Pontifícia Universidade Católica
do Rio de Janeiro

Thales Miranda de Almeida Vieira

Galerias Inteligentes e otimização de posicionamento de câmera

Tese de Doutorado

Tese apresentada ao Programa de Pós–graduação em Matemática Aplicada do Departamento de Matemática da PUC–Rio como requisito parcial para obtenção do título de Doutor em Matemática Aplicada

Orientador : Prof. Thomas Lewiner
Co–Orientador: Prof. Adelailson Peixoto
Co–Orientador: Prof. Luiz Velho

Rio de Janeiro
Janeiro de 2010

Pontifícia Universidade Católica
do Rio de Janeiro

Thales Miranda de Almeida Vieira

Galerias Inteligentes e otimização de posicionamento de câmera

Tese apresentada ao Programa de Pós–graduação em Matemática Aplicada do Departamento de Matemática do Centro Técnico Científico da PUC–Rio como requisito parcial para obtenção do título de Doutor em Matemática Aplicada. Aprovada pela comissão examinadora abaixo assinada.

Prof. Thomas Lewiner
Orientador
Departamento de Matemática — PUC–Rio

Prof. Adelailson Peixoto
Co–Orientador
Instituto de Matemática – UFAL

Prof. Luiz Velho
Co–Orientador
IMPA

Prof. Claudio Esperança
COPPE – UFRJ

Prof. Esteban Clua
Instituto de Computação – UFF

Prof. Ricardo Marroquim
COPPE – UFRJ

Prof. Geovan Tavares
Departamento de Matemática – PUC-Rio

Prof. Sinésio Pesco
Departamento de Matemática – PUC-Rio

Prof. José Eugenio Leal
Coordenador do Centro Técnico Científico — PUC–Rio

Rio de Janeiro, 15 de Janeiro de 2010

Thales Miranda de Almeida Vieira

Graduou–se em Ciência da Computação na Universidade Federal de Alagoas. Concluiu o Mestrado em Matemática na Universidade Federal de Alagoas. Sua dissertação foi intitulada "Registro Automático de Superfícies Usando Spin Images", tendo como área de concentração Computação Gráfica. Desde então, vem trabalhando em diversos temas durante a preparação do doutorado no Laboratório Matmídia da PUC-Rio, incluindo Posicionamento de Câmera, Registro de Superfícies, Parametrização de Superfícies, Deformação de Superfícies e Video 3D, gerando trabalhos publicados. Também participou de dois projetos de pesquisa para a Petrobras: Netgocad e Kernelsis, estando este último diretamente relacionado com o tema desta tese.

Ficha Catalográfica

Vieira, Thales

Galerias Inteligentes e otimização de posicionamento de câmera / Thales Miranda de Almeida Vieira; orientador: Thomas Lewiner; co–orientadores: Adelailson Peixoto, Luiz Velho. — Rio de Janeiro : PUC–Rio, Departamento de Matemática, 2010.

v., 78 f: il. ; 29,7 cm

1. Tese (Doutorado em Matemática Aplicada) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática.

Inclui referências bibliográficas.

1. Matemática – Tese. 2. Galerias Inteligentes. 3. Galerias de Design. 4. Computação Gráfica. 5. Aprendizagem. 6. Posicionamento de Câmera. 7. Algoritmos Genéticos. 8. Máquinas de Suporte Vetorial (SVM). I. Lewiner, Thomas. II. Peixoto, Adelailson. III. Velho, Luiz. IV. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Matemática. V. Título.

CDD: 510

Agradecimentos

A meus pais, João e Magnólia, por todo tipo de apoio;
A meu orientador, Thomas Lewiner, por todo apoio, dedicação, paciência, incentivo e amizade cultivada nesses anos;
A meus co-orientadores Adelailson Peixoto e Luiz Velho, por todo apoio e pelas discussões produtivas;
À Sara, pelo carinho, paciência, incentivo e perseverança nestes anos;
A todos os professores do Laboratório Matmídia, especialmente os professores Hélio Lopes e Geovan Tavares pelas colaborações no desenvolvimento deste trabalho;
A meus colegas Alex Bordignon e Rener Castro pelas colaborações no desenvolvimento deste trabalho;
A todos os meus colegas do Laboratório Matmídia pela amizade e por todo e qualquer tipo de colaboração nesses anos;
A todos os funcionários do Departamento de Matemática da PUC-Rio;
À PUC-Rio, pela bolsa de isenção VRAC;
À CAPES, pela bolsa de doutorado.

Resumo

Vieira, Thales; Lewiner, Thomas; Peixoto, Adelailson; Velho, Luiz. **Galerias Inteligentes e otimização de posicionamento de câmera**. Rio de Janeiro, 2010. 78p. Tese de Doutorado — Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

A Computação Visual levanta vários problemas envolvendo um grande número de parâmetros e tendo uma forte componente subjetiva. Um exemplo de tal problema é o do fotógrafo, tentando otimizar a posição, orientação e abertura da câmera para capturar uma cena tridimensional. As Galerias de Design são comumente usadas em tais problemas para apoiar o usuário na busca da melhor combinação de parâmetros. Porém, a escolha repetida dessas combinações de parâmetros em várias instâncias do problema torna as Galerias de Design usuais laboriosas e não aproveita as interações passadas do usuário para ajudá-lo. Este trabalho apresenta as Galerias Inteligentes, uma nova abordagem para realizar aprendizagem de problemas subjetivos, tais como posicionamento de câmera. A interação do usuário com Galerias de Design ensina uma máquina de aprendizagem estatística. A máquina treinada é capaz de imitar o usuário, classificando parâmetros ou selecionando automaticamente o melhor parâmetro. O processo de aprendizagem baseia-se nas Máquinas de Suporte Vetorial (SVM), que realiza a classificação de parâmetros representados por uma coleção de descritores. Realizamos experiências com o problema de otimização de posicionamento de câmera, que demonstraram que a técnica proposta é eficiente e lida com cenas apresentando elevada oclusão e complexidade de profundidade. Este trabalho também inclui validações de usuário da interface da galeria inteligente.

Palavras–chave

Galerias Inteligentes. Galerias de Design. Computação Gráfica. Aprendizagem. Posicionamento de Câmera. Algoritmos Genéticos. Máquinas de Suporte Vetorial (SVM).

Abstract

Vieira, Thales; Lewiner, Thomas; Peixoto, Adelailson; Velho, Luiz. **Intelligent Galleries and Camera Positioning Optimization**. Rio de Janeiro, 2010. 78p. Tese de Doutorado — Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

In Visual Computing applications, several problems arise that involve a large number of parameters and have a strong subjective component. An example is the photographer problem, which tries to optimize the position, orientation and field of view to capture a tridimensional scene. Design Galleries are commonly used to support the user in such multi-parameter optimization. However, repeatedly choosing parameters for several instances of a problem turns usual Design Galleries labourious and does not use previous user interactions to help him. This work presents Intelligent Galleries, a learning approach for subjective problems such as the camera placement. The interaction of the user with a design gallery teaches a statistical learning machine. The trained machine can then imitate the user, either by classifying parameters or by automatically searching the best one. The learning process relies on a Support Vector Machines for classifying views from a collection of descriptors. We perform experiments with the automatic camera placement, which demonstrate that the proposed technique is efficient and handles scenes with occlusion and high depth complexities. This work also includes user validations of the intelligent gallery interface.

Keywords

Intelligent Galleries. Design Galleries. Computer Graphics. Learning. Camera Positioning. Genetic Algorithms. Support Vector Machines (SVM).

Sumário

Lista de figuras

Lista de tabelas

Sumário das notações

Símbolo	Significado
Ω	Espaço de parâmetros
n	Dimensão do espaço de parâmetros Ω
p_k	Dimensão do espaço do descritor k
p	Dimensão do espaço de descritores concatenados
m	Quantidade de descritores
Ψ	Conjunto de amostras no espaço de parâmetros
ω_i	Componente do parâmetro i
Π	Espaço de resultados
π	Funções geradoras de resultados
φ_j	Descritor j
φ	Aplicação que concatena descritores
x_i	Vetor de descritores $\varphi(\omega_i)$
c_i	Classificação do parâmetro ω_i
Γ	Banco de dados de treinamento
q_v	Ponto v de um objeto
$\mathcal{L}$	Operador de Laplace-Beltrami discreto
κ_H	Operador de Curvatura Média
$\mathcal{S}$	Operador de Saliência
$\mathcal{V}_i$	Conjunto de pixels não nulos da imagem $\pi(\omega_i)$
pix	Pixel de uma imagem
K	*Kernel*
w	Vetor normal do hiperplano classificador das Máquinas de Suporte Vetorial
b	Vetor de deslocamento do hiperplano classificador das Máquinas de Suporte Vetorial
α_i	Vetor de suporte das Máquinas de Suporte Vetorial
ξ_i	Variável *slack* das Máquinas de Suporte Vetorial
Φ	Conjunto de descritores de parâmetros
χ	Função de normalização de descritores
x_i^l	Coordenada l do vetor de descritores $x_i = \varphi(\omega_i)$
g_l	Gráfico bidimensional de pontos do tipo $(x_i^l, \hat{f}(x_i))$
φ^l	Componente l da aplicação φ
ξ_i	Variáveis *slack* das Máquinas de Suporte Vetorial
C	Peso associado à flexibilidade das Máquinas de Suporte Vetorial
ϕ	Função não-linear do classificador não-linear
$\mathcal{F}$	*Feature space* das Máquinas de Suporte Vetorial
$\hat{f}$	Função linear em $\mathcal{F}$
ν	Quantidade de parâmetros selecionados em uma galeria para reprodução genética
w^p	Posição da câmera
w^d	Direção de visão da câmera
w^o	Orientação da câmera

Prefácio

Comecei a trabalhar na área de Computação Visual em 2005, quando ingressei no Programa de Mestrado em Matemática da Universidade Federal em Alagoas, logo após receber o grau de Bacharel em Ciência da Computação. Durante meu mestrado, fui orientado pelo Prof. Dr. Adelailson Peixoto, que me direcionou para a área de Modelagem Geométrica. Nesta fase, estudei o problema de Registro de Superfícies, o qual foi tema de minha dissertação intitulada "Registro Automático de Superfícies Usando Spin-Images" (47).

Em seguida, optei por ingressar no Programa de Pós-Graduação do Departamento de Matemática da Pontifícia Universidade Católica do Rio de Janeiro para realizar meu doutoramento, sob orientação do Prof. Dr. Thomas Lewiner. Tive a oportunidade de participar de dois projetos de pesquisa para a Petrobras, com meu colega Alex Bordignon, intitulados NetGoCad e Kernelsis-br. Algumas técnicas que usamos neste trabalho foram aplicadas no projeto Kernelsis-br na área de análise das Máquinas de Suporte Vetorial.

Na área acadêmica, continuei trabalhando com Registro e Reconstrução de Superfícies, tendo publicado dois trabalhos em conferências da área, intitulados "*An iterative framework for registration with reconstruction*" (50) e "*Geometry super-resolution by example*" (48), além de um trabalho sobre união de bolas intitulado "*Scale-Space for Union of 3D Balls*" (6).

Esta tese é fruto de um trabalho realizado, desde o começo de meu doutoramento em Matemática, três anos atrás, com colaborações de meu colega Alex Bordignon, que resultou na publicação do artigo "*Learning Good Views Through Intelligent Galleries*"(49), o qual apresentamos no *Eurographics 2009* e foi publicado no *Computer Graphics Forum*.

1
Introdução

1.1
Motivação

Nos últimos anos, foi marcante a evolução do poder de processamento gráfico e o aumento da capacidade de armazenamento dos computadores, além do surgimento e popularização de dispositivos gráficos para aquisição de imagens e geometria, como câmeras digitais e *scanners* 3D, por exemplo.

Consequentemente, novos problemas na área de Computação Visual surgiram ou tornaram-se mais complexos. Profissionais das mais diversas áreas, como engenheiros, publicitários e artistas, além de usuários inexperientes, passaram a interagir diretamente com o computador para tratar dados cada vez maiores e de origens diferentes.

Para resolver tais problemas, os usuários precisam ajustar parâmetros cada vez mais complexos e de alta dimensão. Podemos dar como exemplo: ajuste de parâmetros de câmera e iluminação em cenas tridimensionais; ajuste de parâmetros de *rendering* para dados volumétricos; edição de imagens e vídeos; e alinhamento de *scans*.

Quando o problema tem soluções objetivas, é possível recorrer a métodos automáticos para otimizar parâmetros. Porém, em problemas subjetivos, o usuário precisa interagir para expressar suas preferências. Em geral, este trabalho é realizado pelo método de tentativa e erro, tornando-se muitas vezes trabalhoso e não intuitivo.

As Galerias de Design (26) se apresentam como uma abordagem alternativa para facilitar o trabalho do usuário na busca do parâmetro perfeito. Em suas interfaces, um conjunto de parâmetros é apresentado para o usuário na forma de um mosaico de imagens ou animações, como ilustra a figura 1.1. Posteriormente, os parâmetros selecionados pelo usuário neste mosaico são usados para gerar novos mosaicos, ou galerias, refinando a seleção inicial (ver seção 2.1). Porém, as Galerias de Design não aproveitam os parâmetros selecionados pelo usuário, que formam um rico banco de dados com suas pre-

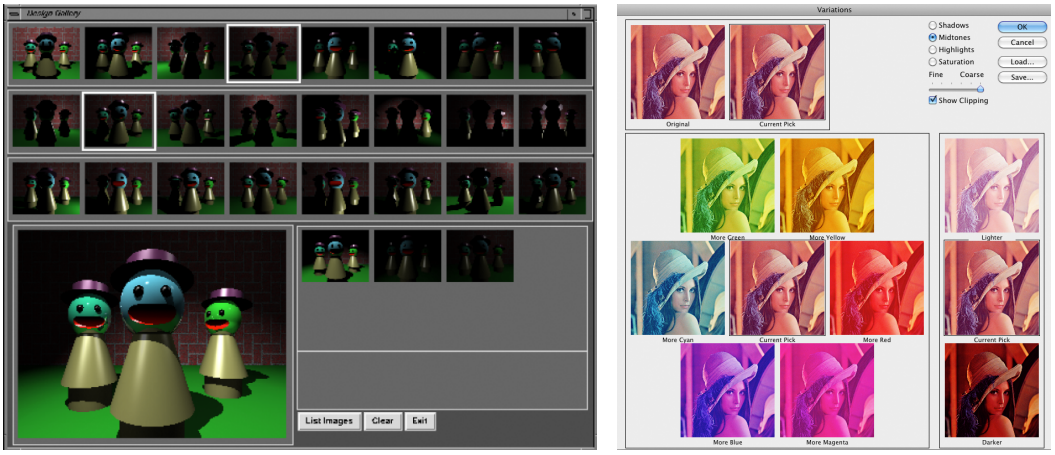

Figura 1.1: Exemplos de Galerias de Design para posicionar luz em cenas tridimensionais (esquerda); e para recolorir imagens (direita)

ferências, para facilitar seu trabalho em interações futuras. Consequentemente, a interação do usuário com as Galerias de Design torna-se laboriosa e repetitiva quando é necessário realizar ajustes de parâmetros em várias instâncias de um problema.

Apresentamos neste trabalho as Galerias Inteligentes, uma nova abordagem para realizar aprendizagem de problemas subjetivos em Computação Visual. Em nosso método, a interação do usuário com as Galerias de Design é usada para ensinar uma máquina de aprendizagem estatística. Consequentemente, esta máquina torna-se capaz de reproduzir suas preferências e participar do processo de ajuste de parâmetros, facilitando o trabalho do usuário.

Usaremos como aplicação principal para as Galerias Inteligentes um problema de extrema importância para a Computação Visual: o problema do fotógrafo, ou problema de otimização de posicionamento de câmera em cenas tridimensionais. Em particular, trata-se de um problema que tem uma natureza fortemente subjetiva, pois depende das preferências do usuário e da aplicação.

1.2 Trabalhos relacionados às Galerias de Design

As Galerias de Design foram apresentadas em 1997 por Marks *et al.* (26) com o objetivo de ajudar o usuário a explorar espaços de parâmetros multidimensionais através de representações visuais. No trabalho original, são descritos usos das galerias em diversos problemas não lineares, como: seleção e posicionamento de luz em cenas tridimensionais; seleção de funções de transferência de cor e opacidade para renderização de volumes; além de aplicações em animação.

O uso de Galerias de Design se disseminou rapidamente na indústria

e no meio acadêmico. Na indústria, é possível encontrar Galerias de Design em *softwares* populares como o Adobe After Effects CS4 (42), onde um componente denominado *Brainstorm* usa uma interface baseada nas Galerias de Design para ajuste de parâmetros de animação. No Adobe Photoshop CS4 (43), Galerias de Design são usadas para recolorir imagens, exibindo variações de cores da imagem em uma interface de galerias. A figura 1.1 contém dois exemplos de uso de galerias: o primeiro foi extraído do artigo original de Marks *et al.*, e trata do problema de posicionamento de luz em cenas tridimensionais; o segundo é uma captura de tela do *software* Adobe Photoshop (43), ilustrando o uso de galerias para recolorir imagens.

No meio acadêmico, galerias já eram usadas para realizar ajuste de parâmetros antes mesmo do trabalho original de Marks *et al.*. Em 1996, He *et al.* (18) trataram do problema de otimização de funções de transferência para visualização de dados volumétricos. No método proposto, o usuário seleciona parâmetros organizados em galerias de imagens, e métodos estocásticos usam as seleções para gerar novos parâmetros.

Diversos outros trabalhos se basearam em galerias para resolver problemas de ajuste de parâmetros multidimensionais. Entre os mais recentes, destacamos o trabalho de Ngan *et al.* (29), que usam galerias para selecionar parâmetros de funções de distribuição de reflectância bidirecional (BRDF); e o trabalho de Shapira *et al.* (39), onde galerias são usadas para recolorir imagens, explorando um espaço de alta dimensão representando possíveis manipulações de cor.

Finalmente, na área de visualização e exploração de dados, diversas abordagens tentam representar na tela bidimensional do computador dados multidimensionais. Em 2001, Jankun-Kelly e Ma (19) usam uma estratégia semelhante às Galerias de Design para explorar espaços de parâmetros multidimensionais, gerando visualizações que dependem da navegação do usuário diretamente no espaço de parâmetros. Lim *et al.* (25) propuseram um método para visualizar dados biomédicos que tenta organizar em um *layout* bidimensional dados multidimensionais, preservando ao máximo a distância original entre os pontos do espaço multidimensional. Recentemente, Bordignon *et al.* (5) propuseram um protótipo de interface para explorar visualmente dados multidimensionais, baseando-se em coordenadas estelares.

1.3 Trabalhos relacionados à otimização de posicionamento de câmera

O problema de otimização de posicionamento de câmera tem recebido muita atenção nas últimas décadas (10). Segundo Christie *et al.* (9), esses métodos podem ser classificados como métodos de maximização de visibilidade, otimização de descritores de visão e regras de design.

Maximização de visibilidade Provavelmente, a primeira tentativa de calcular boas visões foi de Kamada e Kawai (20), onde os autores consideram ótimo o ponto de vista que minimiza o número de faces degeneradas em uma projeção ortogonal. Aqui, entende-se por degeneradas arestas que são projetadas em pontos ou polígonos projetados em retas. Este trabalho foi estendido posteriormente para projeções perspectivas no trabalho de Gómez *et al.* (14). Note que estes trabalhos não levam em conta a oclusão das faces.

Em seguida, critérios de visibilidade bidimensionais foram introduzidos por Plemenos and Benayada (33). Nesse trabalho, foi desenvolvida uma medida que leva em conta a área projetada e o número de triângulos visíveis, removendo as regiões oclusas da superfície. Estendendo esse trabalho, Vázquez *et al.* (46) definiu uma medida chamada entropia de ponto de vista, que mede a qualidade da visão como a quantidade de informação que esta provê para o observador, levando em conta, em cada face, a fração de sua área projetada visível relativa à área total visível. Porém, em cenas de terreno, onde os vetores normais da superfície são similares, essa entropia perde o sentido. Para tratar estes casos, Stoev e Straßer (41) adicionaram um novo termo na função de custo que mede a profundidade máxima da cena.

Definições mais complexas de importância de regiões de superfície foram propostas baseadas em aprendizagem de saliências (21) e busca em bancos de dados (40).

Descritores de visão Em 2005, Lee *et al.* (23) introduziu uma medida baseada em curvatura em multi-escala para medir saliências em malhas. Neste mesmo trabalho, eles consideraram bons pontos de vista como aqueles que maximizavam a saliência de regiões visíveis, realizando uma otimização com gradientes descendentes.

No mesmo ano, Polonsky *et al.* (34) apresentou e comparou descritores de visão 2D e 3D, como área da superfície projetada, curvatura, silhueta e complexidade topológica. Neste trabalho, esses descritores eram maximizados individualmente. Porém, os autores recomendam como trabalhos futuros a

combinação de descritores, possivelmente usando algum procedimento de aprendizagem.

As Galerias Inteligentes auxiliam o usuário em problemas de ajuste de parâmetros realizando aprendizagem em conjuntos de descritores. Consequentemente, apresentam-se como uma alternativa natural para resolver problemas de otimização de posicionamento de câmera.

Regras de design Trabalhos relacionados que usam critérios artísticos e de design são relevantes em especial para contextos de animação. Blanz *et al.* (4) estudaram fatores que influenciam visões de objetos 3D preferidas de humanos, destacando que a relevância de cada fator depende da aplicação, geometria do objeto e familiaridade com o objeto. Gooch *et al.* (15) desenvolveram um sistema baseado nesses fatores e heurísticas usadas por artistas. Para animações, regras de cinematografia têm sido propostas para gerar boas trajetórias de câmera (2, 3, 12, 16, 17).

Apesar de se tratarem de critérios importantes e úteis para posicionamento de câmera, nosso foco é aprender as preferências do usuário para criar regras subjetivas, sem levar em conta ainda regras de design objetivas. Porém, a combinação de regras subjetivas do usuário com regras objetivas é uma possível extensão de nosso trabalho que trataremos adiante.

2
Visão Geral das Galerias Inteligentes

Neste capítulo descrevemos as Galerias Inteligentes e enfatizamos sua aplicabilidade em diversos problemas em Computação Visual. Começaremos descrevendo as Galerias de Design, que são interfaces para gerar parâmetros como, por exemplo, posição de câmera. Em seguida, mostramos como as Galerias Inteligentes incorporam aprendizagem nas Galerias de Design, com o objetivo de classificar e selecionar parâmetros de acordo com as preferências subjetivas do usuário. Finalmente, tratamos da aplicabilidade desta nova abordagem, ilustrando como as Galerias Inteligentes podem ser utilizadas em diversos contextos.

2.1
Galerias de Design

As Galerias de Design foram introduzidas para ajudar o usuário em problemas de Computação Visual que envolvem o ajuste de um grande número de parâmetros para se obter um resultado satisfatório. Podemos dar como exemplos posicionamento de câmera e luz, extração de isosuperfícies a partir de dados volumétricos, funções de transferência de cor para renderização de volumes, entre outros. A otimização dos parâmetros desses problemas é muitas vezes subjetiva e trabalhosa. Um problema que tratamos com mais detalhe nesta tese é do problema do fotógrafo, que consiste em escolher a melhor posição de câmera.

Seja $\Omega \subset \mathbb{R}^n$ o espaço dos parâmetros ω (e.g. a posição da câmera), e Π o espaço dos resultados, tipicamente imagens ou representações gráficas. É importante destacar que, em geral, os parâmetros ω são multi-dimensionais, o que complica a busca de soluções. Ademais, a função $\pi\colon \Omega \mapsto \Pi$ que mapeia parâmetros em resultados é muitas vezes não linear e descontínua, dificultando o trabalho do usuário. Isso é o caso do *rendering*, onde Ω é um espaço que pode incluir, por exemplo, a posição da câmera e sua direção de visão. Além disso, a avaliação de π por si só pode ser computacionalmente cara.

As Galerias de Design (26) foram introduzidas em 1997 com o objetivo

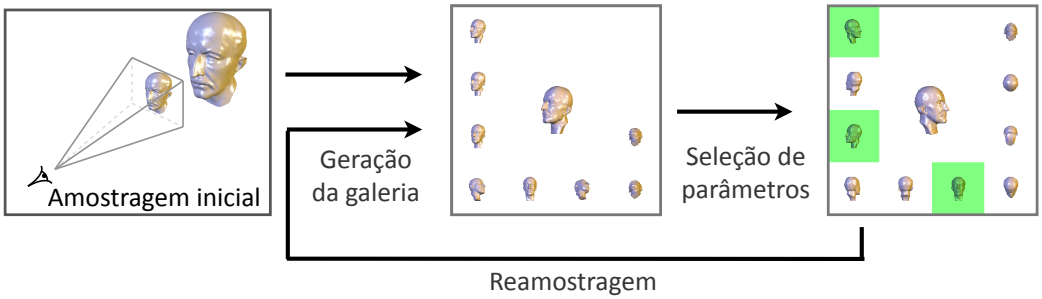

Figura 2.1: Galerias de Design: uma galeria é gerada a partir de uma amostragem $\Psi \subset \Omega$. O usuário seleciona bons parâmetros e opcionalmente uma nova galeria é gerada com parâmetros de Ω que refinam as seleções do usuário.

de dar assistência ao usuário na seleção de parâmetros quando o resultado puder ser representado por meio de imagens, ou seja, quando Π for um espaço de imagens. Nesta metodologia, realiza-se inicialmente uma amostragem do espaço dos parâmetros $\Psi \subset \Omega$. O conjunto de amostras é dado de entrada para a função π, e finalmente o conjunto das imagens resultantes $\pi(\Psi)$ é organizado em uma interface onde o usuário pode interagir selecionando parâmetros. Opcionalmente, uma reamostragem no espaço dos parâmetros é realizada considerando-se a seleção do usuário e seguindo-se alguma estratégia de refinamento. No trabalho original (26), este refinamento é realizado por *clustering*. A figura 2.1 descreve as Galerias de Design.

2.2 Galerias Inteligentes

Podemos observar que os parâmetros selecionados pelo usuário nas suas interações com Galerias de Design formam um rico banco de dados que caracteriza suas preferências. Desse ponto de vista, um método que seja capaz de interpretar esse banco de dados será capaz também de aprender as preferências do usuário, e poderá ajudá-lo a resolver problemas de ajuste de parâmetros que respeitem suas preferências subjetivas.

Com este objetivo, as Galerias Inteligentes estendem o uso das Galerias de Design realizando aprendizagem supervisionada, ou seja, aprendizagem por exemplo, a partir deste banco de dados. Em particular, esta abordagem é apropriada para resolver problemas subjetivos. A figura 2.2 exibe os principais componentes das Galerias Inteligentes.

Na nova abordagem, os parâmetros são classificados pelo usuário como bons ($c_i = +1$) ou ruins ($c_i = -1$). Esta classificação é processada e incor-

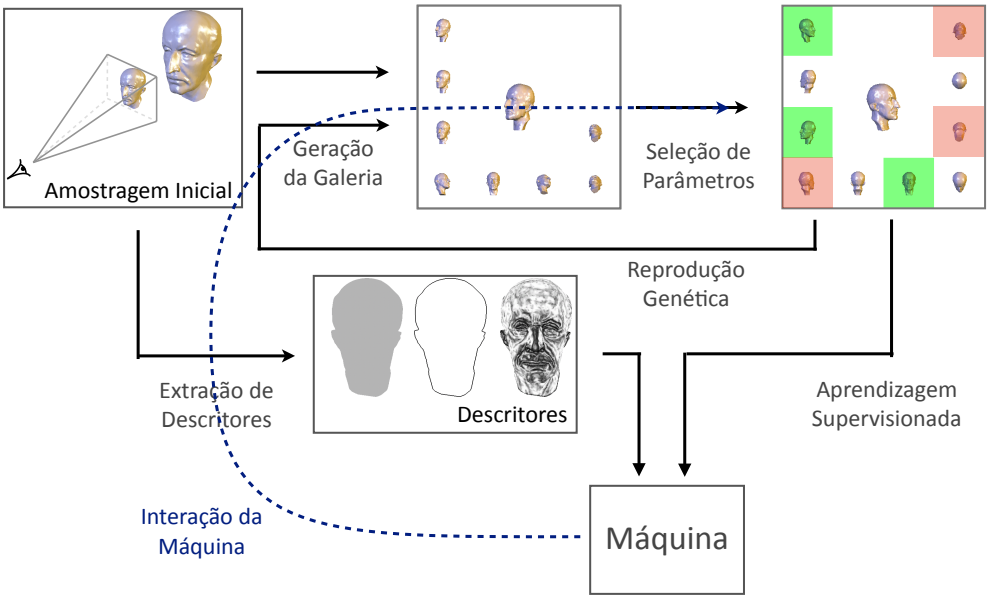

Figura 2.2: Galerias Inteligentes: uma galeria é gerada a partir de uma amostragem $\Psi \subset \Omega$. O usuário classifica imagens $\pi(\omega_i) \in \pi(\Psi)$ como boas (verde) ou ruins (vermelho), ou simplesmente ignora (branco). Os parâmetros selecionados são incorporados a uma máquina de aprendizagem e usados posteriormente para ajudar o usuário a escolher parâmetros.

porada a um banco de dados de treinamento (*training set*). O processamento dos parâmetros classificados pelo usuário envolve a extração de descritores, que representam as preferências do usuário de maneira mais explícita. Observe que o próprio parâmetro ω_i geralmente não pode ser usado como descritor, pois pode não ser intrínseco da instância do problema. A etapa de extração de descritores será explicada detalhadamente no capítulo 5.

A partir das seleções do usuário, uma nova galeria é gerada, refinando a seleção do usuário na galeria anterior. Essa geração pode ser feita por um algoritmo genético que realiza combinações convexas no espaço dos parâmetros (ver apêndice B). Porém, outras estratégias de refinamento poderiam ser utilizadas.

Usando o banco de dados atualizado, a máquina de aprendizagem é capaz de imitar o usuário, ordenando a galeria, colocando bons parâmetros em posições privilegiadas, ou realizando uma seleção automática dos melhores parâmetros. Para esta função, usamos neste trabalho uma adaptação das Máquinas de Suporte Vetorial (45). Justificamos sua escolha pelas suas carac-

terísticas e qualidades que atendem aos requisitos de nossa abordagem. Estas serão explicitadas mais adiante.

Uma introdução às Máquinas de Suporte Vetorial é dada no apêndice A. A adaptação desenvolvida neste trabalho é apresentada no capítulo 3 e analisada no capítulo 4.

2.3 Aplicabilidade

As Galerias Inteligentes se apresentam como uma boa alternativa para resolver problemas subjetivos em Computação Visual, devido a sua capacidade de aprender preferências do usuário por meio de exemplos. Mencionamos aqui alguns destes problemas e desenvolvemos um em particular no final deste trabalho.

Exemplos de problemas nos quais as Galerias Inteligentes se aplicam incluem:

1. Edição de imagens: As Galerias Inteligentes podem ser aplicadas para aprender preferências envolvendo diversos parâmetros de edição de imagens, como brilho, contraste, saturação, parâmetros específicos de filtros, entre outros.

2. Extração de isosuperfícies: Um problema bastante relevante em Computação Gráfica trata de descobrir os melhores isovalores de uma função definida sobre um volume com o objetivo de extrair uma isosuperfície.

3. Iluminação de cenas tridimensionais: Dada uma cena tridimensional, as Galerias Inteligentes podem ser utilizadas para aprender e aplicar parâmetros de luz como posição, direção, intensidade e tipo de luz.

4. Funções de transferência em volumes: As Galerias Inteligentes podem ser utilizadas para, dada uma função de densidade definida sobre um volume, determinar funções de transferência de cor para renderização do volume, provendo um melhor entendimento do objeto gráfico (35, 27, 31).

5. Otimização de posicionamento de câmera (problema do fotógrafo): Dado um modelo tridimensional, podemos usar as Galerias Inteligentes para resolver o problema de escolher parâmetros de câmera que renderizem uma imagem bidimensional ótima. No caso de animações tridimensionais, como em simulação de multidões (30) e jogos (13) , este problema torna-se mais complexo e laborioso, pois é necessário escolher parâmetros de câmera em cada quadro da animação, respeitando a coerência temporal.

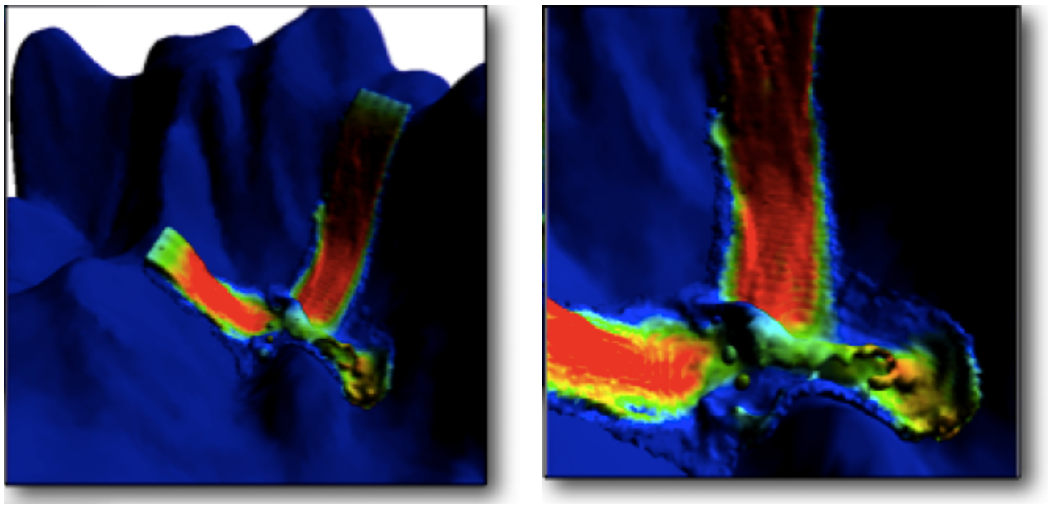

Figura 2.3: Subjetividade: melhor visão de um quadro de uma simulação de fluídos de acordo com as preferências de um designer (esquerda) e de um pesquisador na área (direita).

Trataremos neste trabalho especialmente de otimização de posicionamento de câmera em cenas tridimensionais. Este problema é altamente subjetivo, pois cada usuário possui suas próprias preferências, que podem mudar também de acordo com a aplicação, como ilustrado na figura 2.3. Outra característica importante é sua natureza não-linear: pequenas alterações de parâmetros, como a posição da câmera, podem alterar completamente a avaliação do usuário sobre a visão resultante. Isto justifica o uso de aprendizagem não linear, usando Máquinas de Suporte Vetorial, por exemplo.

Neste problema, o espaço de parâmetros Ω é composto pelo espaço da câmera, o qual é multidimensional, em geral. Dependendo da aplicação, este espaço pode ser composto não somente pela posição da câmera, mas também pela direção de visão, orientação da câmera (*up vector*), entre outros. Em nosso trabalho, dependendo da aplicação, alguns parâmetros de câmera são mantidos fixos para diminuir a dimensão do problema. A função de mapeamento π será simplesmente uma função de *rendering*, que recebe atributos da câmera e renderiza uma imagem contida no espaço de resultados Π. Esta função pode ser extremamente descontínua, principalmente em cenas com grande potencial de oclusão.

Com estas características, o problema de encontrar bons pontos de vista mostra-se apropriado para ser explorado com Galerias Inteligentes. A figura 2.4 exibe um exemplo de galeria gerada a partir de uma amostragem uniforme em um espaço de câmera. A partir desta galeria inicial, o usuário seleciona os parâmetros, ou pontos de vista, classificando-os como bons ou ruins (verde e vermelho respectivamente na figura 2.4), ou ignorando-os. A partir desta

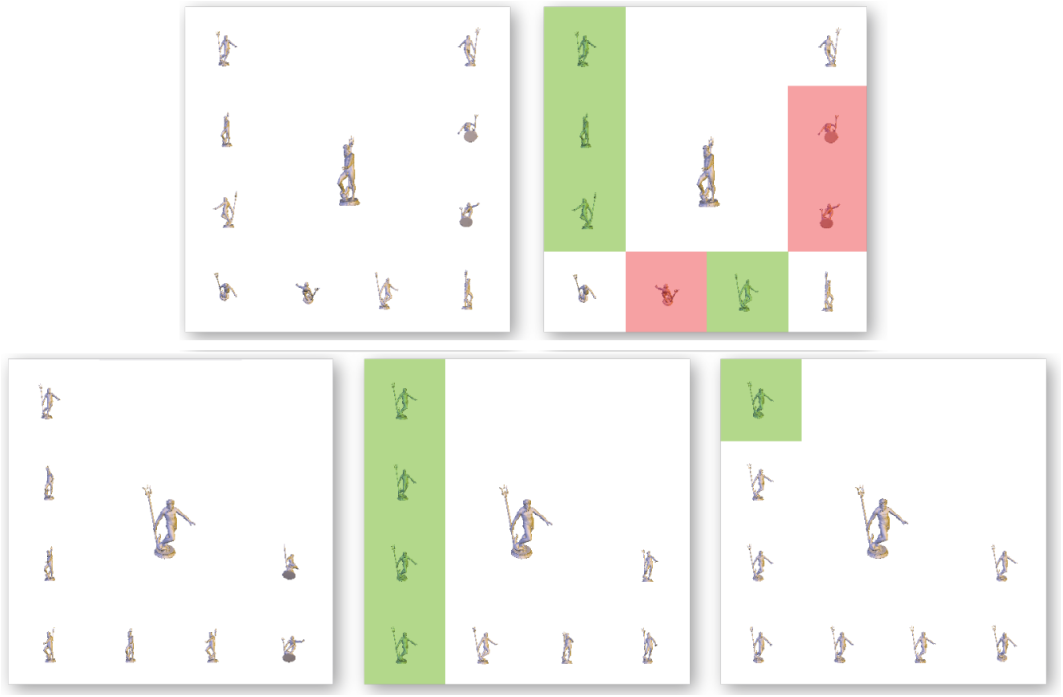

Figura 2.4: Exemplo do uso das Galerias Inteligentes: a partir de uma galeria inicial (superior esquerda), o usuário seleciona bons e maus parâmetros (superior direita); uma nova galeria é gerada (inferior esquerda) e o usuário continua o processo (inferior meio) até chegar a uma visão final satisfatória (inferior direita).

seleção, uma nova galeria é gerada usando um algoritmo genético. A máquina então usa as seleções do usuário para aprender suas preferências, e em seguida ordena e seleciona as melhores visões. Este processo continua até que o usuário encontre uma visão final satisfatória.

Finalmente, o uso das Galerias Inteligentes pode acelerar bastante processos como, por exemplo, determinar as melhores visões em todos os quadros chaves de uma animação. Neste caso, o usuário poderia selecionar visões de alguns *keyframes* usando as Galerias Inteligentes, que em seguida seriam capazes de selecionar automaticamente as visões dos *keyframes* subsequentes. Experimentos desta natureza são realizados e apresentados no capítulo 6.

3
Aprendizagem

Neste capítulo descrevemos como as Galerias Inteligentes usam a interação do usuário com sua interface para aprender suas preferências, e como esse conhecimento é aplicado para ajudar o usuário.

3.1
Máquinas de aprendizagem

No capítulo 2, apresentamos uma visão geral das Galerias Inteligentes como uma nova abordagem que usa parâmetros selecionados pelo usuário em uma interface de galerias para aprender suas preferências. Isto caracteriza o problema de aprendizagem supervisionada, ou seja, aprendizagem por exemplo. Nessa abordagem, é necessário criar um banco de dados Γ (*training set*) que caracterize cada parâmetro ω_i na forma de um vetor $x_i \in \mathbb{R}^p$ (*input*) associado a uma classificação c_i (*output*), que no nosso caso representa um bom ou mau parâmetro, de acordo com a avaliação do usuário.

Para incluir um parâmetro ω_i no banco de dados Γ, é necessário extrair descritores φ_j que representem suas principais características usando números. Formalmente, dado um parâmetro $\omega_i \in \Omega$, cada descritor será representado como uma aplicação $\varphi_j \colon \Omega \mapsto \mathbb{R}^{p_j}$, que avalia parâmetros e retorna vetores de números reais. Estes vetores são concatenados em um único vetor $x_i \in \mathbb{R}^p$, tal que $x_i = \varphi(\omega_i) = (\varphi_1(\omega_i), \varphi_2(\omega_i), \ldots, \varphi_m(\omega_i))$, onde $p = \sum_{k=1}^{m} p_k$.

Além disso, representamos a classificação binária do usuário (como boa ou ruim) usando uma variável $c_i \in \{-1, 1\}$. Os pares $(\varphi(\omega_i), c_i)$ vão representar o dado associado a ω_i no banco de dados Γ, que deve ser utilizado por algum método que realize aprendizagem supervisionada.

Neste trabalho, usamos uma derivação das Máquinas de Suporte Vetorial, uma classe de métodos que nos possibilita realizar classificação não linear. Dentre suas qualidades, destacamos:

– Eficiência computacional: O cálculo da função classificadora e avaliação é eficiente e paralelizável;

– Boa adaptação em bancos de dados esparsos: apenas dados na fronteira das classes, ou vetores de suporte, são considerados;

– Complexidade computacional pouco dependente da dimensão do espaço de entrada $\mathbb{R}^p$: o cálculo da função classificadora e sua avaliação tem sua complexidade dominada pela quantidade de amostras do banco de dados, o que nos permite adicionar novos descritores na formulação do problema sem prejudicar consideravelmente a eficiência;

– Classificação robusta a dados imprecisos: em diversos cenários, o usuário pode ter dificuldade em classificar parâmetros em regiões próximas à fronteira das classes. Usando uma formulação com margem flexível, as Máquinas de Suporte Vetorial consideram apenas dados distantes da fronteira para realizar classificações;

– Matematicamente bem fundamentada: Ótima pela teoria de aprendizagem estatística Vapnik-Chervonenkis (45);

O apêndice A contém uma introdução sobre esses métodos.

Dado um banco de dados representado por um conjunto

$$\Gamma = \{(x_i, c_i) | x_i \in \mathbb{R}^p, c_i \in \{-1, 1\}\}_{i=1}^{n}, \tag{3-1}$$

desejamos inicialmente criar um classificador não linear $g: \mathbb{R}^p \mapsto \{-1, 1\}$ que, dado um parâmetro ω_i representado por seu vetor de descritores $x_i = \varphi(\omega_i)$, nos retorne uma classificação $g(x_i)$ que imite as preferências do usuário, de acordo com o banco de dados. Usando a formulação não linear com margem flexível das Máquinas de Suporte Vetorial descrita no apêndice A, criamos um classificador não linear $g: x \mapsto sign\left(\hat{f}(x)\right)$, onde $\hat{f}$ é uma função linear em um espaço de alta dimensão, ou dimensão infinita $\mathcal{F}$:

$$\hat{f}(x) = \sum_{i \in SV} [\alpha_i c_i \langle \phi(x_i), \phi(x) \rangle] + b. \tag{3-2}$$

Aqui, a função $\phi: \mathbb{R}^p \mapsto \mathcal{F}$ mapeia não linearmente os descritores para o espaço $\mathcal{F}$, e é representada implicitamente por *kernels* do tipo $K: \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$. Os valores α_i são multiplicadores de Lagrange e b está relacionado com o deslocamento do hiperplano de margem máxima em relação à origem.

Para escolher o *kernel* mais eficiente para nosso método, poderíamos comparar a eficiência de diversos *kernels*. Porém, optamos, por simplicidade, pelo *kernel* Gaussiano, dado por

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right). \tag{3-3}$$

Para expandir as funcionalidades das Galerias Inteligentes, desejamos também que a função classificadora nos possibilite ordenar parâmetros, do melhor para o pior, respeitando as preferências do usuário. Porém, a função de classificação g retorna apenas um valor binário representando a classe à qual um dado parâmetro pertence.

Para obter uma avaliação contínua consideramos a seguinte idéia: a distância dos descritores $x_i = \varphi(\omega_i)$ ao hiperplano $\hat{f}(x) = 0$ nos dá uma boa aproximação da qualidade do parâmetro ω_i. Por exemplo, pontos distantes do hiperplano devem representar parâmetros muito bons ou muito ruins, de acordo com o sub-espaço ao qual pertencem. Baseando-se nessa intuição, usaremos para ordenação de parâmetros a função $\hat{f}$ (sem o operador sinal).

3.2 Usos das máquinas de aprendizagem nas Galerias Inteligentes

Apresentaremos agora os principais usos das Galerias Inteligentes relacionadas com sua máquina de aprendizagem. Destacaremos seis usos.

Aprendizagem das preferências do usuário A interação da máquina de aprendizagem com a interface das Galerias Inteligentes começa no momento em que o usuário seleciona imagens $\pi(\omega_i)$ na galeria, na fase que chamaremos de treino. Neste cenário, o usuário pode classificar parâmetros como bons ou ruins, ou simplesmente ignorá-los, como na figura 3.1. Cada parâmetro ω_i selecionado pelo usuário vai dar origem a um novo par $(\varphi(\omega_i), c_i)$ que será incluído no banco de dados.

Uma possível variação do método aceitaria classificações contínuas por parte do usuário ($c_i \in [-1, 1]$), desde que se realizem adaptações na interface

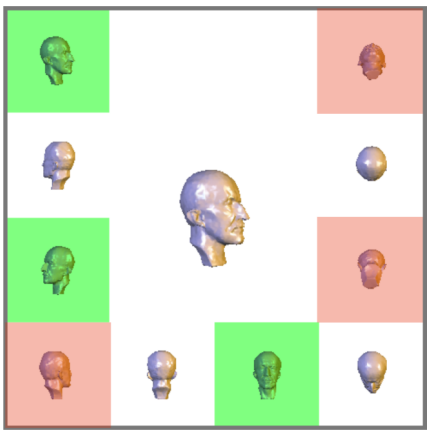

Figura 3.1: Classificação de parâmetros: o usuário pode classificar parâmetros como bons (verdes), ruins (vermelhos) ou ignorá-los.

das galerias que não prejudiquem a experiência do usuário. Além disso, a possibilidade de ignorar parâmetros da galeria simplifica o trabalho do usuário, que pode se focar em classificar apenas parâmetros que considere relevantes ou cuja classificação seja mais clara para ele, tornando desnecessária a classificação de todos os parâmetros da galeria.

A partir do momento em que o banco de dados é alimentado com dados provenientes das seleções do usuário, a máquina torna-se capaz de imitar suas preferências, realizando seleções e ordenações de parâmetros com o objetivo de ajudar o usuário. Essa aprendizagem pode ser analisada com o objetivo de destacar os descritores mais usados pelo usuário, como proposto no capítulo 4.

Seleção automática de bons parâmetros Este procedimento permite mostrar as escolhas da máquina ao usuário. As seleções automáticas podem ser executadas quando uma nova galeria é gerada, pelo método de amostragem inicial do domínio Ω, ou como resultado de combinações das seleções do usuário. A máquina gera sua função classificadora $\hat{f}$ usando o banco de dados atualizado, e seleciona parâmetros classificados como bons por $\hat{f}$. Uma possível variação deste procedimento é a seleção dos ν melhores parâmetros.

Correção de seleções automáticas Devemos considerar que sempre existe a possibilidade de a máquina errar a preferência do usuário, principalmente em treinos iniciais, quando o banco de dados armazena apenas uma ou duas dezenas de parâmetros. Porém, a partir do momento em que a máquina seleciona parâmetros da galeria, o usuário pode corrigir essas seleções, que serão incorporadas ao banco de dados, e, consequentemente, a máquina será capaz de imitar as preferências do usuário com mais precisão.

Destacamos que esse aumento de precisão da função classificadora também é notável em dois cenários: quando o usuário treina galerias mais finas, ou seja, galerias que são criadas após o processo de classificação de parâmetros pelos usuários; e quando o usuário combina o treino de diferentes instâncias como, por exemplo, seleção de parâmetros de câmera de cenas diferentes.

Ordenação dos parâmetros da galeria Outra aplicação importante é a ordenação de parâmetros da galeria. Neste cenário, os parâmetros ω_i da galeria são ordenados de acordo com suas classificações $\hat{f}(\omega_i)$, e reposicionados na interface, destacando sempre os melhores, com o objetivo de facilitar a identificação de bons parâmetros pelo usuário. Isto permite oferecer ao usuário boas opções primeiro, acelerando a sua escolha.

Seleção automática do melhor parâmetro O usuário pode usar este procedimento para obter automaticamente o parâmetro ótimo de acordo com a avaliação da máquina. A seleção automática do melhor parâmetro é realizada por refinamento sucessivo das galerias usando um algoritmo genético. Partindo de uma galeria inicial, a máquina seleciona sozinha os ν melhores parâmetros e gera uma nova galeria, refinando a seleção anterior. A máquina continua selecionando os melhores parâmetros e gerando galerias mais finas, até convergir para uma galeria onde todos os parâmetros são semelhantes.

Usando a linguagem dos algoritmos genéticos, queremos otimizar uma função objetivo representada pela função classificadora $\hat{f}$. A população inicial do algoritmo é representada pelos parâmetros de uma galeria inicial, e as galerias seguintes representam evoluções da população inicial. Em cada geração, os melhores indivíduos (parâmetros) são selecionados para reprodução, onde são recombinados por pares e reproduzidos por combinações convexas. O apêndice B dá uma introdução aos algoritmos genéticos.

Incorporação de regras objetivas Finalmente, em diversos problemas, é necessário combinar regras subjetivas, como as representadas pela função classificadora, com regras objetivas. Por exemplo, no problema de posicionamento de câmera em uma cena tridimensional, a seleção das visões pelo usuário fornece um banco de dados para gerar regras subjetivas. Essas regras poderiam ser combinadas a princípios de cinematografia, por exemplo, para filtar bons parâmetros. Esse tipo de filtragem por regras objetivas pode ser facilmente acoplada nas Galerias Inteligentes, removendo dos parâmetros selecionados os que não satisfazem a regra.

3.3 Componentes das Galerias Inteligentes

Nesta seção vamos descrever formalmente os procedimentos realizados pelas Galerias Inteligentes.

Pré-processamento e geração da galeria inicial Na fase de pré-processamento, a instância do problema é carregada, e as computações necessárias para tornar as próximas etapas eficientes são realizadas, como cálculo de propriedades, por exemplo.

Para gerar a galeria inicial, realizamos uma amostragem uniforme no espaço de parâmetros Ω. O conjunto de amostras $\Psi \subset \Omega$ é utilizado para gerar o conjunto de imagens $\pi(\Psi)$ que irá representar os parâmetros na interface

da galeria. Além disso, nesta fase também é gerado um conjunto $\Phi = \varphi(\Psi)$ contendo os vetores de descritores dos parâmetros.

Para permitir que os dados de treino sejam compatíveis com diferentes instâncias do problema, sempre devemos normalizar os valores dos descritores $x_i = \varphi(\omega_i)$ de cada instância específica do problema. Usamos o conjunto de descritores Φ que acabamos de criar para gerar uma função de normalização $\chi\colon \mathbb{R}^p \mapsto \mathbb{R}^p$, que deve ser aplicada, a partir de agora, em todos os descritores desta instância. Este procedimento de normalização é essencial para a robustez do método e será analisado mais detalhadamente no capítulo 4.

Geração da função classificadora e classificação de parâmetros Os parâmetros classificados pelo usuário em cada galeria treinada são incorporados ao banco de dados Γ em forma de pares $(\chi(\varphi(\omega_i)), c_i)$. O banco de dados atualizado dá origem a uma função de classificação $\hat{f}$ sempre que seja necessário avaliar parâmetros. Em geral, essa etapa ocorre no momento de geração de uma nova galeria, quando um novo conjunto de parâmetros Ψ' é criado e precisa ser classificado.

Reprodução e organização da nova galeria O conjunto de parâmetros classificados pelo usuário é dado como entrada para um algoritmo genético que irá gerar um novo conjunto de parâmetros $\Psi' \subset \Omega$ e dará origem a um novo conjunto de imagens $\pi(\Psi')$ e um novo conjunto de descritores $\Phi' = \chi(\varphi(\Psi'))$.

Em seguida, este conjunto de descritores Φ' é dado como entrada para a função classificadora com o objetivo de ordenar os parâmetros de Ψ' por qualidade, ou seja, do melhor para o pior, de acordo com os valores de $\hat{f}(\Phi')$. Após esta ordenação, as imagens $\pi(\Psi')$ são organizadas na galeria, privilegiando-se os melhores parâmetros. Por simplicidade, organizamos estes parâmetros do melhor para o pior no sentido anti-horário, começando do canto superior esquerdo, como ilustra a figura 3.2.

Seleção automática Em qualquer momento após alimentar o banco de dados pela primeira vez, o usuário pode solicitar ajuda das Galerias Inteligentes para selecionar bons parâmetros na galeria atual. Descrevemos dois métodos para realizar esta seleção.

No primeiro método, a galeria seleciona na interface todos os parâmetros $\omega \in \Psi$, tais que $\hat{f}(\chi(\varphi(\omega))) > 0$. Na segunda estratégia, selecionamos os ν melhores parâmetros, de acordo com as avaliações $\hat{f}(\Phi)$. Note que se

Figura 3.2: Galeria ordenada pela função classificadora do melhor para o pior parâmetro, começando do canto superior esquerdo no sentido anti-horário.

$\lambda = \left|\left\{\omega \in \Psi | \hat{f}(\chi(\varphi(\omega_i))) > 0\right\}\right| < \nu$, então apenas λ parâmetros serão selecionados.

4
Análise

Neste capítulo descrevemos procedimentos para analisar a influência de descritores e otimizar a máquina de aprendizagem. Além disso, descrevemos mais detalhadamente o problema de normalização dos descritores e analisamos parâmetros da interface.

4.1
Análise heurística da influência de descritores

Nesta seção descrevemos um procedimento adotado para analisar a influência individual dos descritores na avaliação da função classificadora. Esta análise tem como objetivo descobrir quais descritores mais influenciam a avaliação dos parâmetros. Em seguida, apresentaremos um método para calibrar parâmetros das Máquinas de Suporte Vetorial usando a influência dos descritores.

Estimativa de influência de descritores Relacionar variáveis de uma função não linear individualmente é uma tarefa difícil. Os métodos existentes para realizar análise não linear, como Kernel PCA (37), não se aplicam em nosso caso, pois procuramos uma dependência entre parâmetros lineares (descritores) e não linear ($\hat{f}$). Por isso resolvemos adotar uma abordagem heurística.

Dado um conjunto de descritores $\Phi = \{x_1, x_2, \ldots, x_k\} \subset \mathbb{R}^p$ e uma função classificadora $\hat{f}$, podemos plotar p gráficos contendo coordenadas dos descritores e a classificação dada por $\hat{f}$, ou seja, cada gráfico g_l^Γ deve conter os pontos $(x_i^l, \hat{f}(x_i))$, onde x_i são pontos de um banco de dados previamente treinado Γ e x_i^l representa a l-ésima coordenada do vetor de descritores $x_i = \varphi(\omega_i)$.

Cada gráfico g_l^Γ pode nos dar uma intuição visual da dependência do classificador $\hat{f}$ com a coordenada l do vetor de descritores. Descritores influentes devem gerar gráficos que apresentem algum padrão, enquanto descritores com menor influência devem gerar gráficos com uma distribuição aleatória dos pontos.

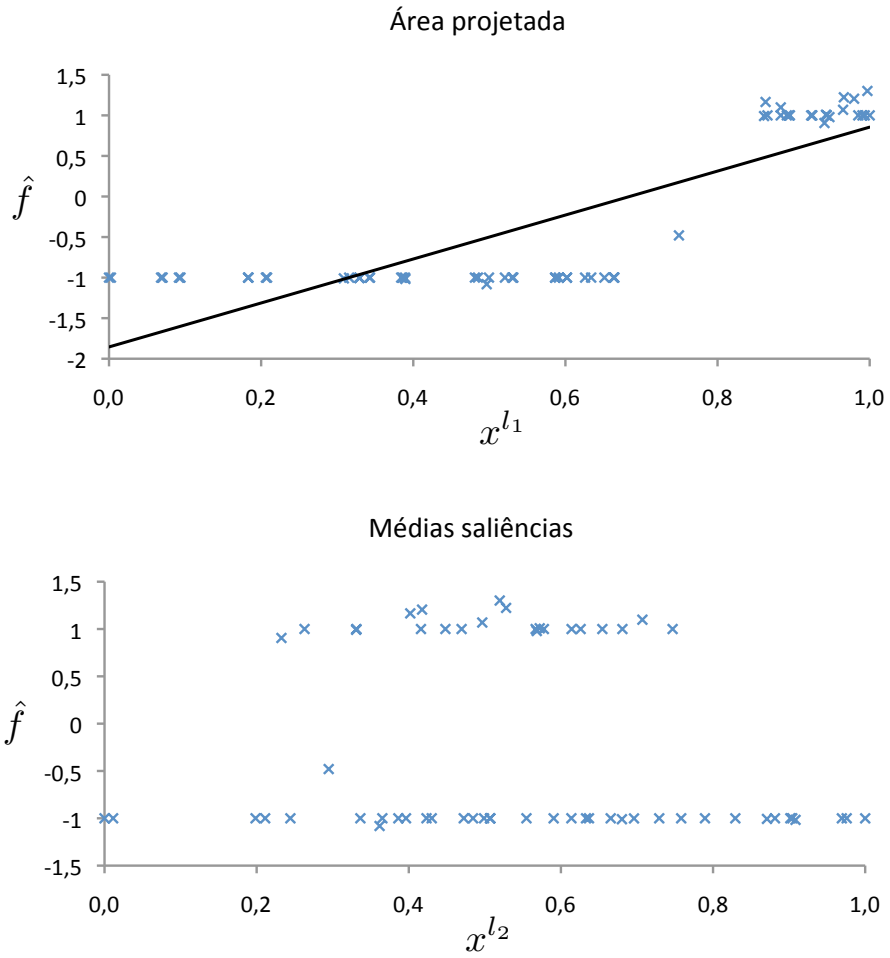

Figura 4.1: Exemplos de gráficos g_l^Γ de influência de descritores: descritor de área projetada na imagem apresentando alta influência, e descritor de médias saliências apresentando baixa influência em um banco de dados obtido através de experiências com um usuário para o problema de otimização de posicionamento de câmera.

Para tentar identificar essas dependências no gráfico, tentaremos aproximar as dependências não lineares com aproximações lineares, ou seja, realizando uma regressão linear em cada gráfico, como exibido na figura 4.1. Para medir a precisão de nossa aproximação, usaremos a variância η_l^Γ dos pontos em relação à reta de regressão. Se esta variância for baixa, a aproximação linear é válida. Consideraremos um descritor influente quando o valor absoluto da inclinação $|\tau_l^\Gamma|$ da reta de regressão for alta e a variância for baixa. Nossa medida de influência é expressa por $|\tau_l^\Gamma|(2 - \eta_l^\Gamma)$. O fator 2 permite balancear heuristicamente estas duas propriedades. O uso do produto em vez da soma ponderada não muda essencialmente o resultado da otimização. A figura 4.1 exibe exemplos de um descritor influente e um descritor pouco influente em um banco de dados obtido a partir de experimentos com um usuário para o

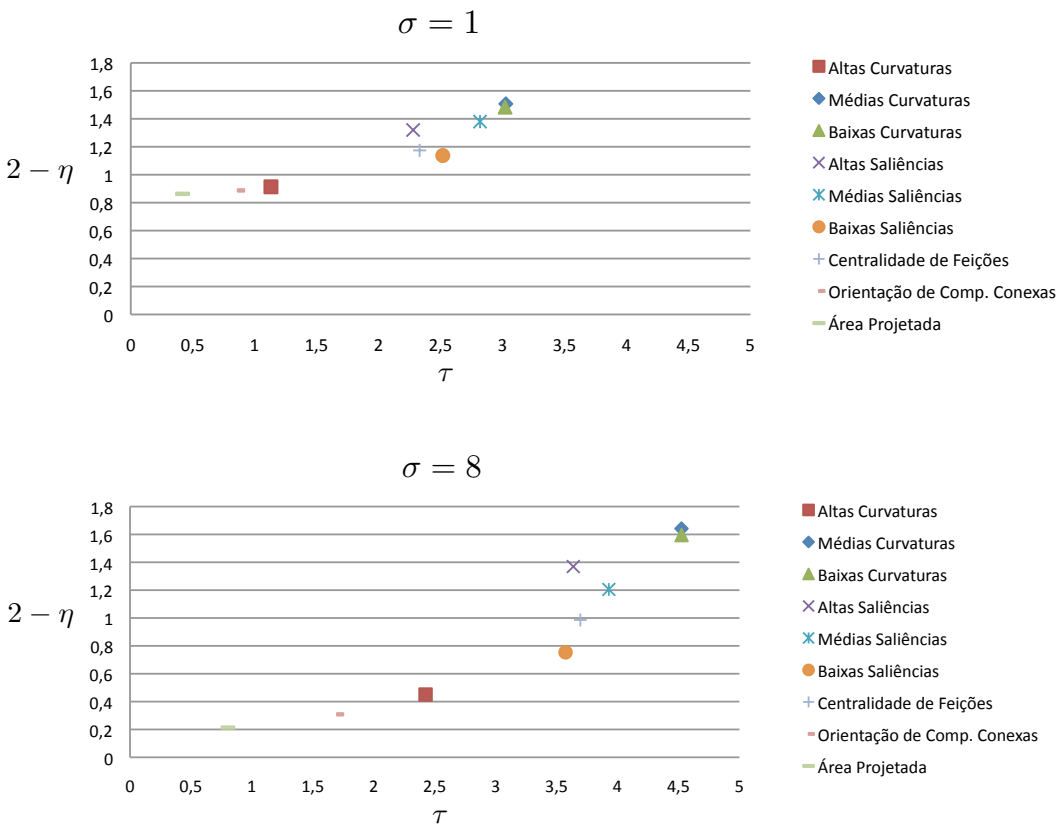

Figura 4.2: Influência de descritores com $\sigma = 1$ (acima) e com $\sigma = 8$ (abaixo). A otimização do parâmetro σ do *kernel* Gaussiano maximiza a influência dos descritores.

problema de otimização de posicionamento de câmera.

Calibração de parâmetros A formulação das Máquinas de Suporte Vetorial define a função classificadora usando *kernels*, que podem depender de um parâmetro, como por exemplo o σ do *kernel* Gaussiano dado na Equação (3-3). Podemos usar o procedimento de análise de influência para otimizar este tipo de parâmetro.

A figura 4.2 exibe exemplos de gráficos representando a influência de descritores do problema de otimização de posicionamento de câmera. Um descritor é influente quando está na região superior direita, que corresponde a valores altos de $|\tau|$ e baixos de η. O gráfico de cima foi obtido usando o *kernel* gaussiano com parâmetro $\sigma = 1$. O gráfico de baixo exibe o resultado com o parâmetro σ otimizado, onde é possível perceber que, em geral, os descritores têm maior influência do que no gráfico inicial.

Propomos um paradigma onde uma função classificadora será robusta quando sofrer bastante influência do maior número de descritores. Na prática, isto significa que o processo de aprendizagem está aproveitando bem os

dados provenientes dos descritores para aprender as preferências do usuário. Portanto, nosso objetivo é maximizar a influência dos descritores, dada por

$$\sum_{k=1}^{p} |\tau_k^\Gamma|(2 - \eta_k^\Gamma),$$

onde $|\tau_k^\Gamma|$ é o valor absoluto da inclinação da reta de regressão do gráfico g_k^Γ, e η_k^Γ sua variância.

Este problema de otimização é resolvido usando o método da bisseção, onde subdividimos progressivamente um intervalo grande de valores de σ até se aproximar de um máximo.

4.2 Normalização

No capítulo 3 descrevemos brevemente um procedimento de normalização que deve ser aplicado em cada instância do problema (a função χ). Esta etapa é fundamental para poder usar a aprendizagem obtida pelos descritores em diferentes instâncias.

Intuitivamente, se para uma dada instância um determinado descritor retorna valores reais no intervalo $[0, 10]$, ou seja $\varphi_j(\Omega) \subset [0, 10]$ e em outra instância este valor oscila no intervalo $[20, 30]$, nossa função classificadora não será consistente em ambos os casos. Porém, é razoável imaginar que, normalizando ambos os intervalos para o intervalo $[0, 1]$, teremos valores correspondentes, do ponto de vista do tipo de informação que o descritor representa.

A etapa de normalização tem como objetivo justamente normalizar todos os intervalos $\mathcal{I}^l = [\inf(\varphi^l(\Omega)), \sup(\varphi^l(\Omega))]$, que variam de acordo com a instância do problema. Para realizar uma normalização linear, precisamos conhecer os extremos destes intervalos.

Como não podemos avaliar os descritores em todo o espaço de parâmetros Ω, determinamos os valores máximos $M = (M^1, M^2, \ldots, M^p)$ e mínimos $\hat{M} = (\hat{M}^1, \hat{M}^2, \ldots, \hat{M}^p)$ que vão definir os intervalos $\mathcal{I}^l$ usando o conjunto de descritores $\varphi(\Psi)$, onde Ψ é a amostragem inicial da galeria. Esta normalização deve ser aplicada sempre que uma nova instância do problema for inicializada na galeria. Note que quanto maior a cardinalidade de Ψ, mais precisos serão os intervalos de normalização.

Após esta etapa, todos os descritores $x_i = \varphi(\omega_i)$ devem ser ser normalizados pela função $\chi\colon \mathbb{R}^p \mapsto \mathbb{R}^p$ dada por

Figura 4.3: Exemplo ilustrando a organização da interface das Galerias Inteligentes: no centro, em alta resolução, o visualizador permite ao observador examinar mais detalhadamente os parâmetros do mosaico.

$$\chi(x) = \left(\frac{x^1 - \inf(\mathcal{I}^1)}{|\mathcal{I}^1|}, \frac{x^2 - \inf(\mathcal{I}^2)}{|\mathcal{I}^2|}, \ldots, \frac{x^p - \inf(\mathcal{I}^p)}{|\mathcal{I}^p|} \right). \tag{4-1}$$

4.3 Otimização da interface

Nesta seção analisamos parâmetros ligados ao algoritmo genético e geração da interface das Galerias Inteligentes.

Nossa interface gráfica é composta de dois tipos de objeto:

- Mosaico de parâmetros: Uma região da interface onde os parâmetros ω_i são representados por suas imagens resultantes $\pi(\omega_i)$ em baixa resolução.
- Visualizador de parâmetros: Uma região onde o usuário visualiza um determinado parâmetro selecionado ω_i através de sua imagem $\pi(\omega_i)$ em alta resolução.

Diversas maneiras de organizar a disposição desses objetos são encontradas na literatura (26, 39). Adotamos uma disposição onde o visualizador é centralizado na interface, e o mosaico compõe a borda da interface, como ilustrado na figura 4.3.

A quantidade de parâmetros de uma galeria influencia as dimensões das imagens no mosaico, garantindo que todos os parâmetros sejam visíveis. Um aspecto a ser analisado é a quantidade de parâmetros ideal em cada galeria, principalmente na inicial. Para cobrir bem o domínio dos parâmetros Ω, a quantidade de amostras do conjunto Ψ deve ser alta. Porém, este excesso de

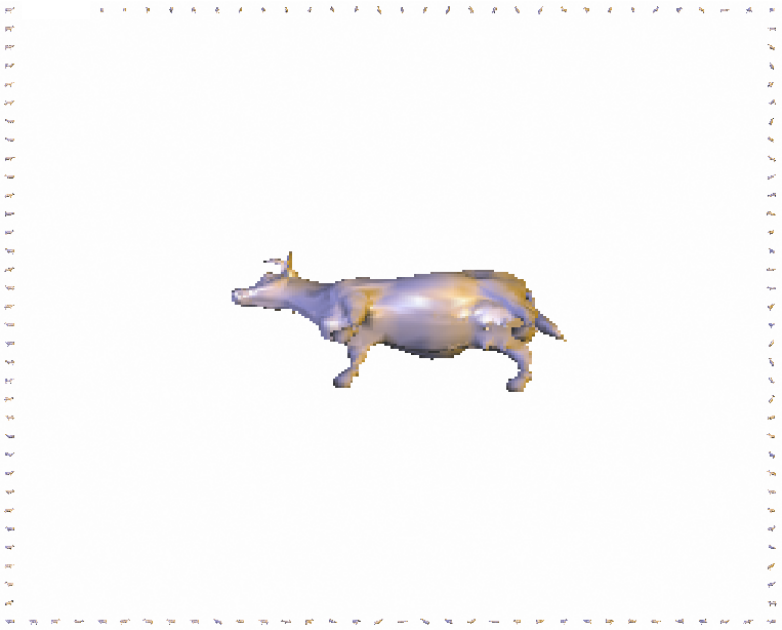

Figura 4.4: Exemplo de galeria com excesso de parâmetros.

parâmetros pode gerar duas inconveniências: o trabalho de classificação torna-se exaustivo para o usuário; e o conseqüente ajuste da resolução das imagens $f(\omega_i)$ do mosaico na interface, tornando-as incompreensíveis pelo usuário, como na figura 4.4.

Uma análise aprofundada desta interface está fora do escopo deste trabalho. Porém, em experimentos simples realizados com usuários, chegamos à conclusão de que galerias com 20 a 36 parâmetros expressam bem a informação para o usuário, além de cobrir razoavelmente bem o espaço de parâmetros em diversos problemas. Eventualmente, os parâmetros podem ser apresentados em galerias sucessivas antes da reprodução. Destacamos que diversos exemplos de galerias, gerados exclusivamente para ilustrar este texto, têm quantidade de parâmetros reduzida para tornarem-se mais visíveis.

A quantidade de parâmetros de cada galeria está diretamente ligada à configuração do algoritmo genético. Em particular, a dimensão de Ω tem que ser pequena para permitir que o algoritmo genético funcione corretamente com poucos parâmetros por galeria (ver seção 7.1). No caso da geração da galeria inicial, esta configuração pode ser ajustada diretamente na fase de geração da população inicial. Nas galerias resultantes de reproduções de parâmetros selecionados pelo usuário, é necessário ajustar a quantidade de pais (seleções do usuário) com a quantidade de filhos gerados por cada par de pais. Este cálculo está descrito na seção B.2 do apêndice B, onde a quantidade de filhos ν de cada pai pode ser ajustada para se aproximar o máximo possível da quantidade de parâmetros ideal da galeria.

5
Descritores para posicionamento de câmera

Neste capítulo focamos no problema de otimização de posicionamento de câmera em cenas tridimensionais. Conforme descrito anteriormente, as Galerias Inteligentes extraem uma coleção de descritores dos parâmetros selecionados. Estes descritores devem representar, usando números reais, características do resultado obtido com um determinado parâmetro. Portanto, é fundamental a escolha de uma coleção de descritores que represente as principais características dos resultados do parâmetro, de modo que a máquina de aprendizagem seja capaz de interpretar e identificar, através de números, as preferências do usuário.

Para otimizar posicionamento de câmera, utilizamos descritores já existentes na literatura, com algumas variações, e desenvolvemos outros para expandir a capacidade de interpretação da máquina para este problema. Utilizamos também descritores para aplicações específicas, como um descritor que mede a velocidade visível em simulação de fluidos. Os descritores selecionados apresentaram resultados satisfatórios em nossos experimentos (ver capítulo 6). Porém, outros descritores podem ser tão ou mais eficientes, em particular se forem consideradas informação de cor, textura e outras características não geométricas. Procuramos apenas colocar o máximo de descritores que foi possível implementar. Esses descritores podem ser classificados de acordo com o domínio em que avaliam os parâmetros:

- Descritores do espaço do objeto: avaliam propriedades (visíveis) dos objetos da cena, como curvatura média e componente conexa;
- Descritores do espaço da imagem: extraem características da imagem renderizada, como a área projetada do objeto na imagem, seu alinhamento em relação aos eixos da imagem e a complexidade da silhueta do objeto.

A primeira seção trata da computação eficiente desses descritores, com o objetivo de possibilitar o uso das Galerias Inteligentes em tempo real ou quasi. Em seguida, descrevemos detalhadamente os descritores utilizados neste

trabalho. Adiantamos que os descritores que se mostraram influentes em boa parte dos experimentos foram os relacionados a propriedades de curvatura e saliência, e o descritor de área projetada.

5.1 Extração eficiente dos descritores

Dado um parâmetro de câmera ω_i, cada descritor φ_j do espaço do objeto realiza um processamento sobre os pontos visíveis da visão resultante $\pi(\omega_i)$, ou seja, na imagem renderizada. Cada descritor está diretamente associado a alguma propriedade dos pontos visíveis, como curvatura média, componente conexa ou *patch*. Essas propriedades são calculadas, normalizadas e quantizadas usando uma determinada quantidade de bits durante a fase de pré-processamento.

Para cada propriedade calculada nos pontos do objeto, geramos e equalizamos seu histograma, para melhorar a distribuição dos valores sobre os objetos. Seja $h_x(a) = \sum_{k=0}^{a} p_x(k)$ a função de distribuição acumulada da propriedade x com probabilidades p_x, de acordo com seu histograma original. Para equalizá-lo, queremos encontrar a transformação $T(x) = y$, tal que a nova função de distribuição acumulada h_y cresça linearmente, ou seja, $h_y(a) \propto a$. É fácil ver que $T(x) = h_x(x)$ respeita nossa condição.

Para acelerar a extração das propriedades dos pontos visíveis, quantizamos estas propriedades dos vértices do objeto e em seguida concatenamos as propriedades quantizadas em cores de 32 bits no formato RGBA. Finalmente, renderizamos uma imagem com as cores codificadas.

A imagem codificada é extraída do *buffer* de cores e dada como entrada para cada descritor, seja ele de atuação no espaço do objeto ou da imagem. Cada descritor saberá decodificar propriedades de interesse nos pixels da imagem. Note que cada pixel não vazio desta imagem corresponde a um ponto visível. Essa estratégia reduz o tempo de processamento dos descritores por imagem, possibilitando a interação em tempo real com as galerias (ver capítulo 6).

A figura 5.2 (esquerda) mostra um exemplo de uma imagem codificada apenas com a curvatura gaussiana, onde os pixels mais escuros representam pontos de alta curvatura. Em nosso caso, foram necessárias 2 imagens codificadas por visão para representar todas as propriedades utilizada com uma quantização adequada. A tabela 5.1 exibe os principais descritores, suas propriedades correspondentes e a quantidade de bits utilizada na codificação.

propriedade	**descritor**	**# bits**
curvatura média	Σ baixas curvaturas visíveis Σ médias curvaturas visíveis Σ altas curvaturas visíveis	6
saliência	Σ baixas saliências visíveis Σ médias saliências visíveis Σ altas saliências visíveis	6
patches	Σ patches visíveis	10
componente conexa	Σ área 2D visível / área 3D da componente conexa Σ área 2D visível × área 3D da componente conexa	10

Tabela 5.1: Propriedades e descritores: cada descritor está associado a uma propriedade dos vértices da cena. Estas propriedades dos vértices são quantizadas usando uma determinada quantidade de bits e codificadas em cores de 32 bits no formato RGBA.

5.2 Descritores do espaço do objeto

Nesta classe de descritores, medimos a visibilidade das propriedades geométricas ou inerentemente tridimensionais dos pontos visíveis, tais como curvatura média, saliência, componentes conexas e *patches*. Desenvolvemos também dois descritores para aplicações específicas: velocidade em simulação de fluidos e visibilidade de grupos em uma animação de batalha.

5.2.1 Descritores geométricos

Boas visões de uma cena tridimensional em geral estão correlacionadas com a quantidade de informação geométrica relevante visível (46, 34, 23). Uma abordagem para medir a quantidade de informação geométrica relevante em uma superfície é baseada na presença de feições (*features*), ou seja, pontos de alta curvatura, como pontos que fazem parte de arestas e quinas. Porém, nem sempre regiões ricas em pontos de alta curvatura são geometricamente importantes, no sentido de representarem partes salientes da geometria do objeto, de acordo com nossa percepção. Um exemplo clássico é exibido na figura 5.1, onde a região do cabelo do David apresenta alta curvatura, porém não se trata de uma região rica em saliências. Por esta razão, utilizamos neste trabalho dois descritores que se complementam, para medir propriedades geométricas de pontos visíveis.

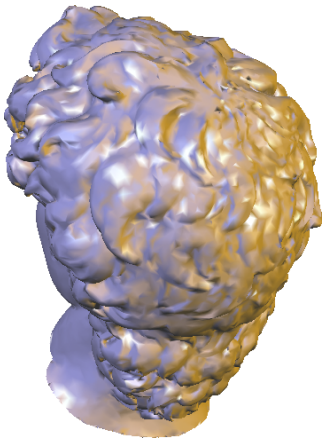

Figura 5.1: Visão traseira da cabeça do David: rica em alta curvatura mas pobre em saliências.

Curvatura O primeiro descritor é baseado na magnitude da curvatura média $|\kappa_H|$ dos pontos visíveis. Durante a fase de pré-processamento, usamos o trabalho de Meyer *et al.* (28) para calcular em cada ponto x_v, o operador de Laplace-Beltrami discreto $\mathcal{L}$ com o peso das cotangentes, dado por:

$$\mathcal{L}(q_v) = \frac{1}{2A_{Mixed}(q_v)} \sum_{q_w \in N_1(q_v)} (\cot \beta_{vw} + \cot \gamma_{vw})(q_v - q_w), \qquad (5\text{-}1)$$

onde $A_{Mixed}(q_v)$ é a região de Voronoi do ponto q_v e $N_1(q_v)$ são os vértices de sua primeira vizinhança estrelada. Daí, a magnitude da curvatura média é dada por $|\kappa_H(q_v)| = \frac{1}{2}\|\mathcal{L}(q_v)\|$. Após o cálculo das curvaturas em todos os pontos, seu histograma é calculado e equalizado.

Saliência O segundo descritor de propriedades geométricas usa uma medida de saliência derivada da curvatura média, apresentada por Lee *et al.* (23). Essa medida de saliência combina suavizações gaussianas da curvatura média em várias escalas ι, dadas pela equação abaixo:

$$\mathcal{S}(q_v, \iota) = \frac{\sum_{x_w \in N(q_v, 2\iota)} \kappa_H(q_w) \exp\left(\frac{-\|q_w - q_v\|^2}{2\iota^2}\right)}{\sum_{q_w \in N(q_v, 2\iota)} \exp\left(\frac{-\|q_w - q_v\|^2}{2\iota^2}\right)}. \qquad (5\text{-}2)$$

Após o cálculo das saliências em cada ponto, seu histograma é calculado e seus valores equalizados. A figura 5.2 (direita) exibe um exemplo de imagem codificada com as saliências, sendo os pontos escuros seus pontos mais salientes.

Para extrair mais características das visões, dividimos essas duas propriedades geométricas em três intervalos: baixos valores $[0, \frac{1}{3})$, médios valores $[\frac{1}{3}, \frac{2}{3})$ e altos valores $[\frac{2}{3}, 1]$. Finalmente, cada descritor soma a quantidade de pixels

Figura 5.2: Descritores geométricos do espaço do objeto: curvatura gaussiana (esquerda) e saliência (direita).

na imagem contendo valores em seu respectivo intervalo, como na Tabela 5.1. Intuitivamente, a máquina poderá aprender se uma dada visão contém grandes (ou poucas) regiões de alta curvatura, média curvatura, baixa curvatura, alta saliência, média saliência e baixa saliência.

Analogamente, tratamos a velocidade dos vértices da malha de um fluido, em um cenário de simulação de fluidos. Neste caso, a velocidade já é um dado de entrada associado a cada vértice, como codificado em cores na figura 2.3.

5.2.2 Descritores de visibilidade de partes

Uma outra linha de descritores do espaço do objeto trata de medir a visibilidade de partes da cena tridimensional. Uma característica importante de uma visão é a área 3D visível. Porém, em algumas aplicações pode ser mais importante manter a maior quantidade de componentes conexas visíveis, independente da área total visível. Em outro cenário, pode ser mais importante ter as componentes conexas de maior área visíveis do que as de menor área, em especial quando a cena tiver grande potencial para oclusões, como ilustrado na figura 5.3.

Pensando nesses problemas, desenvolvemos três descritores, que usam como propriedades um identificador da sua componente conexa, calculado no processo de geração da malha (22), e um identificador de seu *patch*, o qual é gerado na fase de pré-processamento, e descrevemos a seguir.

Seguindo uma estratégia gulosa, inicializamos o *patch* de cada triângulo como ele mesmo. Unimos os patches de menor área usando operações *union-find* (44), de modo a manter os patches com aproximadamente a mesma área, até atingir um determinado limiar. No final do processo, cada triângulo estará associado a um identificador de seu *patch*.

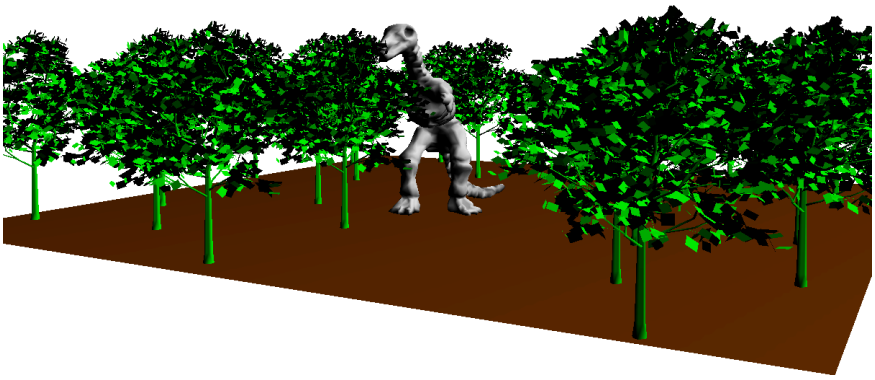

Figura 5.3: Cena rica em oclusão: o dinossauro se esconde atrás das árvores.

Uma outra propriedade que poderia ser utilizada identificaria cada vértice com um segmento da malha. Estes segmentos poderiam ser obtidos usando métodos convencionais de segmentação de malhas (38).

Área 3D visível O primeiro descritor aproxima a área 3D visível medindo a quantidade de diferentes *patches* visíveis na imagem. Como os patches têm aproximadamente a mesma área, contar a quantidade de *patches* visíveis nos dá uma boa aproximação da área 3D visível. É importante notar que se os *patches* gerados forem muito pequenos, corre-se o risco de que a projeção de um *patch* na imagem seja menor que a área de um pixel, e desse modo não seria contabilizado. Além disso, uma grande quantidade de *patches* aumenta a quantidade de bits quantizados na imagem para representá-lo. Por outro lado, *patches* grandes aumentam o erro no cálculo da área 3D causado por *patches* parcialmente visíveis. A figura 5.4 exibe uma imagem codificada com *patches*.

Visibilidade de componentes conexas O segundo descritor desta linha tem o objetivo de medir a quantidade de componentes visíveis, e atua somando a área projetada visível das componentes, normalizada por sua área 3D, como na Tabela 5.1.

Figura 5.4: Descritores de visibilidade de partes do espaço do objeto: *patches* (esquerda) e grupos (direita)

Área relativa Finalmente, para enfatizar grandes componentes, desenvolvemos um descritor que soma a área projetada visível das componentes multiplicada por um peso dado por sua área 3D.

Em uma aplicação específica de animação, usamos como identificador das partes o grupo ao qual vários guerreiros pertencem (figura 5.4)

5.3 Descritores do espaço da imagem

Assim como os descritores do espaço do objeto, os descritores do espaço da imagem usam propriedades dos pontos projetados na imagem. Em particular, serão utilizadas as curvaturas médias e componentes conexas.

Área projetada O primeiro descritor desta linha mede a área projetada no espaço da imagem, somando os pixels com componentes conexas não-nulas.

Silhueta Um outro tipo de característica de uma visão que está relacionada com a percepção do objeto tridimensional pelo observador é a complexidade de sua silhueta. Em geral, silhuetas mais complexas fornecem mais informação ao observador, como ilustrado na figura 5.5, onde é possível identificar claramente a silhueta de um coelho. Definimos o descritor de complexidade da silhueta como a integral do valor absoluto de sua curvatura. Usamos um método simples de continuação para extrair sua silhueta, e somamos os valores absolutos de suas curvaturas em todos os seus pontos pix_t, com peso dado pelo comprimento do segmento $\overline{pix_t pix_{t+1}}$. Usamos o método de Lewiner *et al.* (24) para calcular as curvaturas.

Duas informações importantes de uma visão não foram consideradas até então: a posição dos objetos na imagem, em particular de feições do objeto; e a orientação da imagem.

Centralidade de feições Para obter a primeira informação, criamos um descritor que representa a distribuição dos pontos, dando ênfase aos pontos de alta curvatura média. Este descritor calcula a distância média da posição projetada dos pontos visíveis ao centro da imagem, ponderada por sua curvatura média, ou seja:

$$\varphi_j(\omega_i) = \frac{1}{|\mathcal{V}_i|} \sum_{pix \in \mathcal{V}_i} |\kappa_H(q(pix))| \|pix - center\|,$$

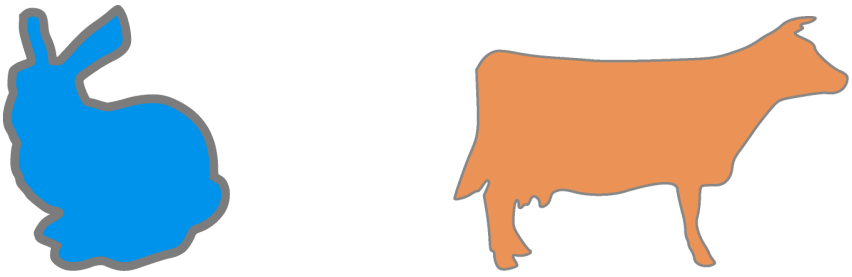

Figura 5.5: Descritores do espaço da imagem: coelho com silhueta complexa (esquerda) e vaca bem orientada em relação aos eixos da imagem (direita).

onde pix é um pixel não nulo da imagem $\pi(\omega)$ representando a projeção de um ponto $q(pix)$ do objeto, $center$ é o centro da imagem e $\mathcal{V}_i$ o conjunto de pixels não nulos da imagem $\pi(\omega_i)$. Este descritor retorna valores baixos se os *features* do objeto estão centralizados na origem, e valores mais altos se estão na borda da imagem.

Orientação das componentes conexas Finalmente, realizamos regressões lineares sobre os pixels correspondentes a cada componente conexo visível na imagem. A inclinação destas retas nos diz se os componentes estão alinhados com os eixos da imagem. Para destacar grandes componentes conexos, este descritor é definido como a média das inclinações destas retas ponderada pelo tamanho de seus respectivos componentes. Na figura 5.5 exibimos um exemplo de uma vaca bem alinhada com os eixos da imagem.

6
Resultados

Neste capítulo apresentamos resultados das Galerias Inteligentes e de sua aplicação ao problema de posicionamento de câmera em cenas tridimensionais usando os descritores apresentados no capítulo 5. Analisamos sua capacidade de aprender e reproduzir as preferências do usuário em diferentes instâncias do problema.

Usamos como instâncias do problema modelos estáticos usuais de Computação Gráfica (ver figuras 6.1, 6.5 e 6.6); modelos de difícil compreensão, como nós tridimensionais (ver figuras 6.8 e 6.9); e cenas com grande potencial de oclusão (ver figuras 6.10 e 6.11).

Usamos como espaço de parâmetros Ω o espaço de câmera composto pela posição da câmera w^p, direção de visão w^d e orientação da câmera w^o. Em nossos experimentos, este espaço foi restrito de acordo com cada aplicação. Analisamos a seguir os resultados.

6.1
Capacidade de aprendizagem e reprodução de preferências

Reprodutibilidade em modelos similares Nosso experimento inicial tem como objetivo verificar se nosso método de aprendizagem combinado com os descritores desenvolvidos para o problema de posicionamento de câmera são capazes de reproduzir definições simples de boas visões. Usamos o modelo de uma vaca como instância de treino, e restringimos a posição da câmera a uma esfera centrada no centro do objeto, e a direção de visão de modo que o observador estivesse sempre visualizando o centro do objeto.

Navegamos pela galeria preferindo sempre visões laterais da vaca (ver figura 6.1). Em seguida, aplicamos o procedimento de seleção automática do melhor parâmetro em modelos de animais similares à vaca, na esperança de obter visões laterais, respeitando o treino.

Os resultados apresentaram-se satisfatórios em todos os casos, inclusive para o cavalo, que tem uma pose um pouco diferente dos demais modelos

Figura 6.1: Posicionamento automático de câmera usando um treino voltado para as visões laterais de uma vaca (esquerda): a máquina é capaz de reproduzir com sucesso visões laterais em outros animais.

(a cabeça não está alinhada com o corpo). Isso mostra que nosso método de aprendizagem é capaz de aprender as preferências subjetivas do usuário e imitar seu comportamento não somente na instância (modelo) treinada, mas também em outras instâncias similares. A escolha dos descritores, fundamental para o sucesso do método, também é validada neste caso: aplicando a análise dos descritores apresentada no capítulo 4, concluimos que os descritores mais relevantes neste caso foram a área projetada, que valoriza visões laterais, e as altas saliências, que evitam visões oblíquas (figura 6.2).

Respeito à subjetividade Gostaríamos também de testar se a máquina é capaz de escolher os parâmetros (visões) levando em conta preferências de diferentes usuários. Nesta análise, pedimos para dois profissionais usarem a galeria para escolher visões em um quadro de uma simulação de fluidos 3D. O primeiro profissional é um *designer* gráfico, que classificou visões do ponto de vista artístico. O segundo é um especialista em dinâmica de fluidos, que posicionou a câmera para visualizar as feições do fluido, gerando visões diferentes das visões do primeiro, como ilustra a figura 6.3.

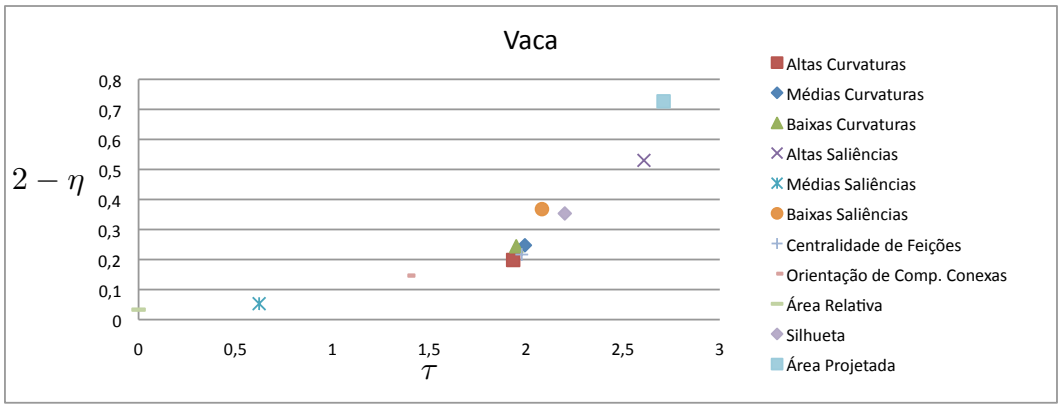

Figura 6.2: Análise da influência dos descritores no treino lateral da vaca (figura 6.1): a região superior direita representa os descritores mais influentes, onde destacam-se a área projetada e as altas saliências.

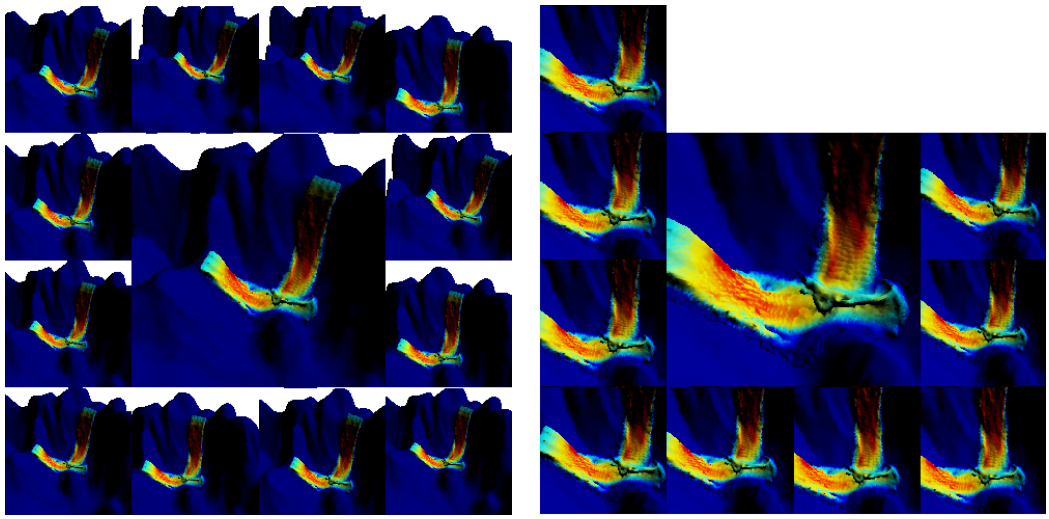

Figura 6.3: Resultado das seleções de visões de um *designer* gráfico (esquerda) e de um especialista em dinâmica de fluidos (direita).

Os bancos de dados Γ^1 e Γ^2 referentes aos treinos dos respectivos profissionais são usados para gerar duas funções classificadoras $\hat{f}^1$ e $\hat{f}^2$. Em seguida, aplicamos o procedimento de seleção automática do melhor parâmetro em outro quadro da simulação usando cada uma das funções, e obtemos resultados similares às preferências de cada usuário, e consequentemente diferentes entre si, como exibido na figura 6.4.

Este resultado valida a capacidade da máquina de respeitar regras subjetivas, gerando resultados diferentes de acordo com as preferências individuais de cada usuário. Além disso, a precisão do método para posicionar a câmera em um quadro diferente da simulação, respeitando o treino em um quadro anterior, nos motiva a realizar experimentos para posicionar câmera em sequências de quadros de video 3D.

Neste experimento, a posição da câmera w^p esteve restrita a um paralelepípedo acima do terreno, e a direção de visão w^d a um cone cujo eixo é perpendicular ao terreno.

Comparação A realização de comparações com outros métodos de posicionamento automático de câmera é delicada, visto que nesses casos são levadas em conta regras objetivas, enquanto nosso método é capaz de fornecer diferentes resultados de acordo com o treino do usuário.

Porém, dependendo do contexto, é possível encontrar boas visões canônicas de modelos, ou seja, visões que são indiscutivelmente ótimas, como a visão frontal da face em um modelo de uma cabeça. Seguindo esta linha, realizamos um treino duplo: primeiro navegamos na galeria selecionando visões

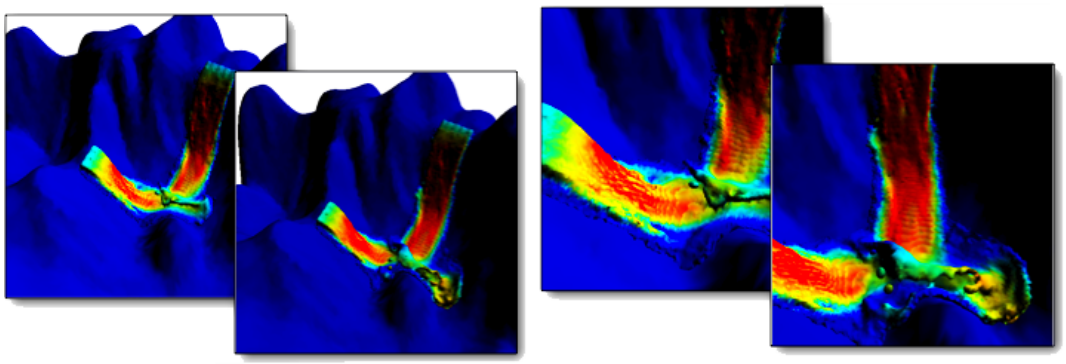

Figura 6.4: Comparação entre as melhores visões escolhidas pelos profissionais em um quadro da simulação de fluidos e as visões resultantes do procedimento de seleção automática em um outro quadro da simulação: a máquina imita com sucesso as preferências do *designer* gráfico (imagens à esquerda) e do especialista em dinâmica de fluidos (imagens à direita), selecionando visões semelhantes às selecionadas por cada profissional.

frontais do rosto do modelo da cabeça do David; e aumentamos a precisão do treino selecionando visões frontais do rosto do Max Planck, como mostra a figura 6.5. Neste caso, o banco de dados resultante é a união das seleções dos dois modelos.

Usando este treino, comparamos nosso método com três regras objetivas descritas em trabalhos recentes: maximização da área projetada (34), maximização do comprimento da silhueta (34) e maximização das saliências (23). Selecionamos alguns modelos com visões canônicas para testar a robustez dos métodos, e destacamos três resultados, exibidos na figura 6.6, onde as regras objetivas falharam em achar a melhor visão, enquanto nosso método teve sucesso, usando os dados do treino duplo.

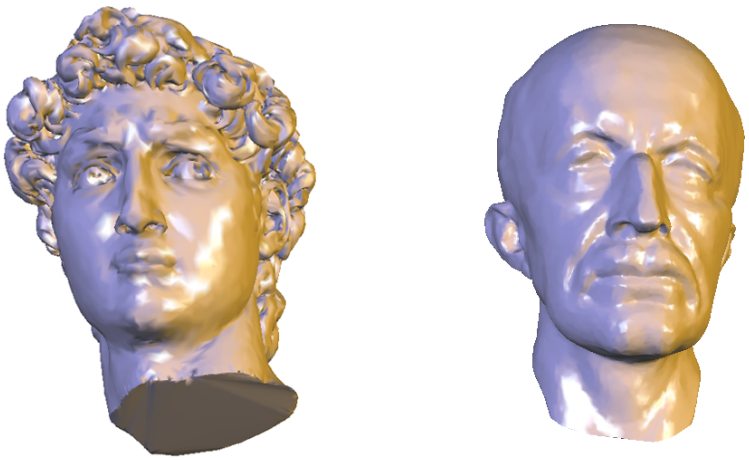

Figura 6.5: Visões preferidas em um treino iniciado na cabeça do David (esquerda) e completada com a cabeça do Max Planck (direita).

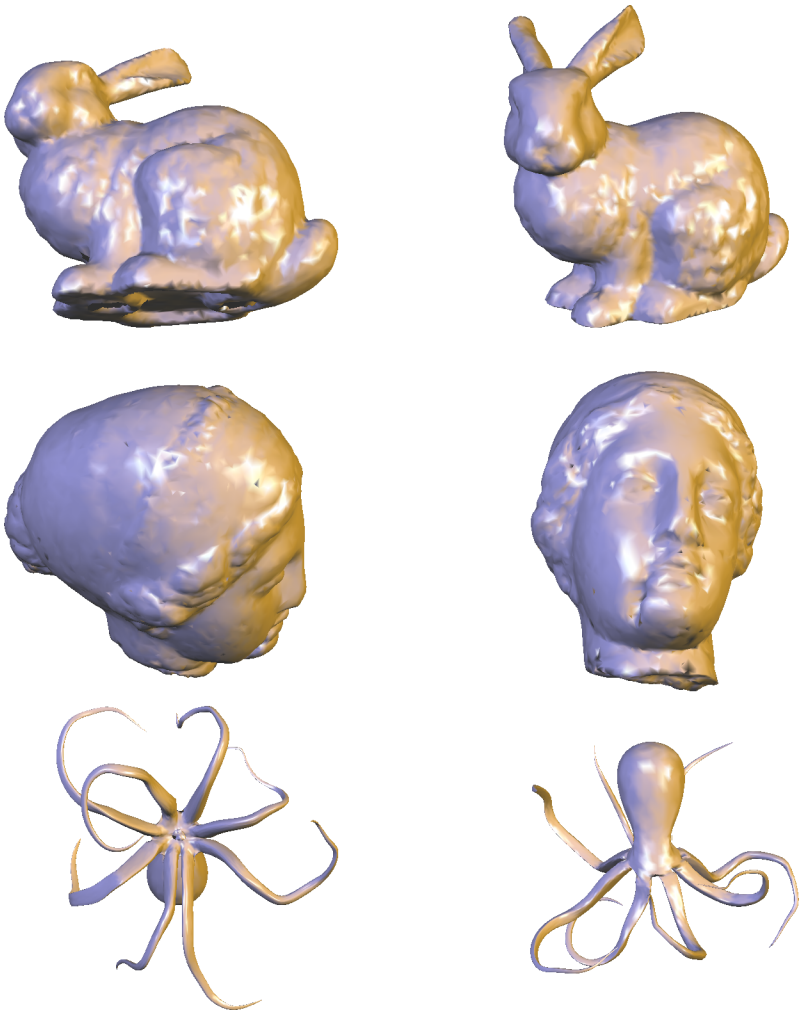

Figura 6.6: Comparação de nosso método de posicionamento automática de câmera (coluna da direita) usando os dados de treino da figura 6.5 com outros trabalhos recentes: maximização das saliências (23) (superior esquerda), maximização da área projetada (34) (meio esquerda) e maximização do comprimento da silhueta (34) (inferior esquerda).

Esses resultados ilustram bem as dificuldades do problema de posicionamento de câmera, onde é difícil definir um conjunto de regras objetivas que funcione bem em todos os casos. Além disso, a natureza subjetiva do problema impossibilita a definição de uma visão canônica em todos os modelos e aplicações. Portanto, trata-se de um problema onde as Galerias Inteligentes se apresentam como uma boa alternativa, aprendendo regras subjetivas através da combinação de diversos descritores, e mostrando-se robusta em uma maior coleção de modelos.

O resultado da análise de influência dos descritores neste treino duplo é exibida na figura 6.7, onde é possível identificar 6 descritores com boa influência, com destaque para curvaturas médias e baixas, e saliências médias. Este resultado nos leva a concluir que a combinação de vários descritores

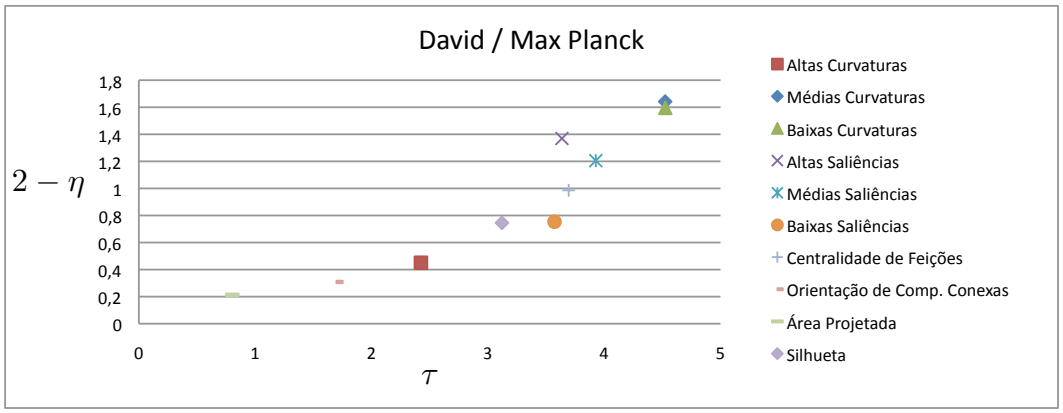

Figura 6.7: Análise da influência dos descritores no treino frontal duplo do David e do Max Planck (figura 6.5): seis descritores se destacam na região superior direita, com destaque para curvaturas médias e baixas, e saliências médias.

é fundamental para gerar uma função classificadora robusta para todos os modelos testados, como predito na conclusão do trabalho.

Neste experimento, o espaço de câmera foi o mesmo usado no experimento que avaliou a reprodutibilidade em modelos similares.

Estabilidade da função classificadora Examinamos também a estabilidade da aprendizagem realizando testes de precisão *out-of-sample*. Removemos aleatoriamente 20% das amostras de Γ dos dados de treinamento, geramos uma função classificadora sem essas amostras e testamos se a função classificadora é capaz de avaliar corretamente as amostras removidas. Realizamos 10 experimentos usando os dados de treino da vaca (figura 6.1), e 10 experimentos usando os dados do treino duplo (figura 6.5). As médias de classificações erradas são exibidas na tabela 6.1.

O baixo índice de erros nos leva a crer que nossa máquina de aprendizagem em combinação com o conjunto de descritores é estável do ponto de vista de não depender de um subconjunto pequeno de amostras de seu banco de dados Γ para gerar funções classificadoras robustas.

Instâncias difíceis Realizamos experimentos em cenas tridimensionais complexas, incluindo nós tridimensionais e cenas com grande potencial de oclusão.

Treino	**Falsos positivos**	**Falsos negativos**
vaca	5.7%	0.2%
duplo	2.7%	3.0%

Tabela 6.1: Estabilidade da função classificadora: baixo índice de erros.

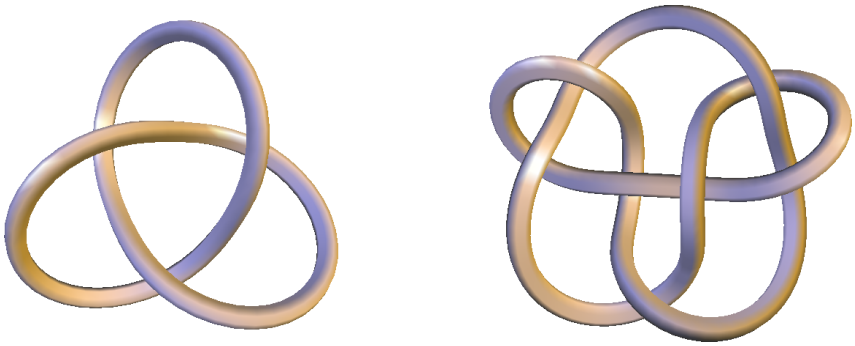

Figura 6.8: Treino do nó trifólio (esquerda) e posicionamento automático da câmera no nó 7.7c (direita): neste treino, foram selecionadas visões descritivas do nó, convergindo para a visão da esquerda. A melhor visão do nó 7.7c segundo a máquina reproduz a preferência do usuário por visões descritivas (direita).

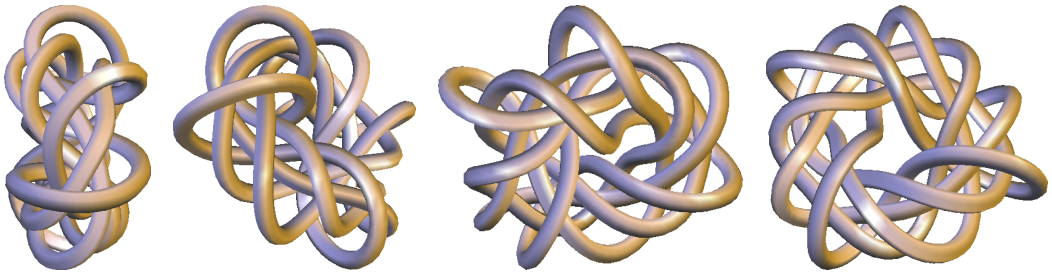

Figura 6.9: Classificação coerente de visões do nó 7.7c (ordem crescente de qualidade de visão da esquerda para direita).

A compreensão de nós tridimensionais por meio de imagens em geral é difícil, sendo necessário encontrar pontos de vista específicos que gerem imagens com boa descrição do nó. Realizamos um treino em um nó simples (trifólio), classificando como boas as visões mais descritivas do nó (figura 6.8), e pedimos para a máquina usar este treino para avaliar visões em nós mais complexos. Vale notar que o mergulho dos nós já resulta de uma otimização de forma (36) independente do uso das Galerias Inteligentes.

A figura 6.8 também exibe o melhor ponto de vista do nó 7.7c de acordo com a avaliação da máquina. Na figura 6.9 exibimos o resultado de uma ordenação de diferentes pontos de vista em um nó mais complexo, de acordo com a avaliação da máquina treinada no nó mais simples. É possível ver claramente que a máquina tem sucesso em classificar coerentemente as visões de acordo com as preferências do usuário, ou seja, visões mais descritivas do nó receberam melhor avaliação.

Neste experimento, o espaço de câmera foi o mesmo usado no experimento que avaliou a reprodutibilidade em modelos similares.

Para testar a capacidade do método em cenas com grande potencial de oclusão, criamos diversos modelos de florestas com árvores posicionadas

Figura 6.10: Treino em cena de floresta com grande potencial de oclusão, na tentativa de visualizar o dinossauro. Melhor visão do usuário (esquerda) e visão de cima da floresta, com a respectiva posição e direção da câmera (direita).

em posições aleatórias, e com um dinossauro posicionado em algum lugar aleatório da floresta. A posição da câmera w^p ficou restrita a um anel fora da floresta, contido no plano da floresta. A direção de visão w^d foi restrita a uma circunferência paralela ao plano da floresta.

Realizamos um treino em uma floresta com pouca densidade de árvores dando preferência aos pontos de vista onde seria possível visualizar a frente do dinossauro sem a oclusão das árvores, como ilustrado na figura 6.10. Usamos este treino para selecionar automaticamente pontos de vista em cenas com maior densidade de árvores. Os resultados podem ser observados na figura 6.11.

A máquina claramente tem sucesso em encontrar pontos de vista onde o dinossauro é visível, inclusive quando os pontos de vista que geram visões satisfatórias são escassos, como no exemplo inferior da figura 6.11. A análise de influência dos descritores revela grande influência do descritor de área relativa. A figura 6.12 exibe o gráfico deste descritor, revelando visualmente alta inclinação da reta e baixa variância.

6.2 Eficiência da interface das Galerias Inteligentes

Para analisar a eficiência da interface, realizamos uma pré-avaliação simples com usuários inexperientes, comparando a interface das Galerias Inteligentes com uma interface clássica, no caso o *Blender* com *trackball*. Apesar de não ser uma interface intuitiva, ela contém as ferramentas usuais de interação 3D. Uma avaliação mais adequada está ainda por ser feita mas sairia do escopo deste trabalho.

Inicialmente, os usuários escolhem um grupo de modelos para avaliar com ambas as interfaces, usando apenas uma das interfaces em cada modelo

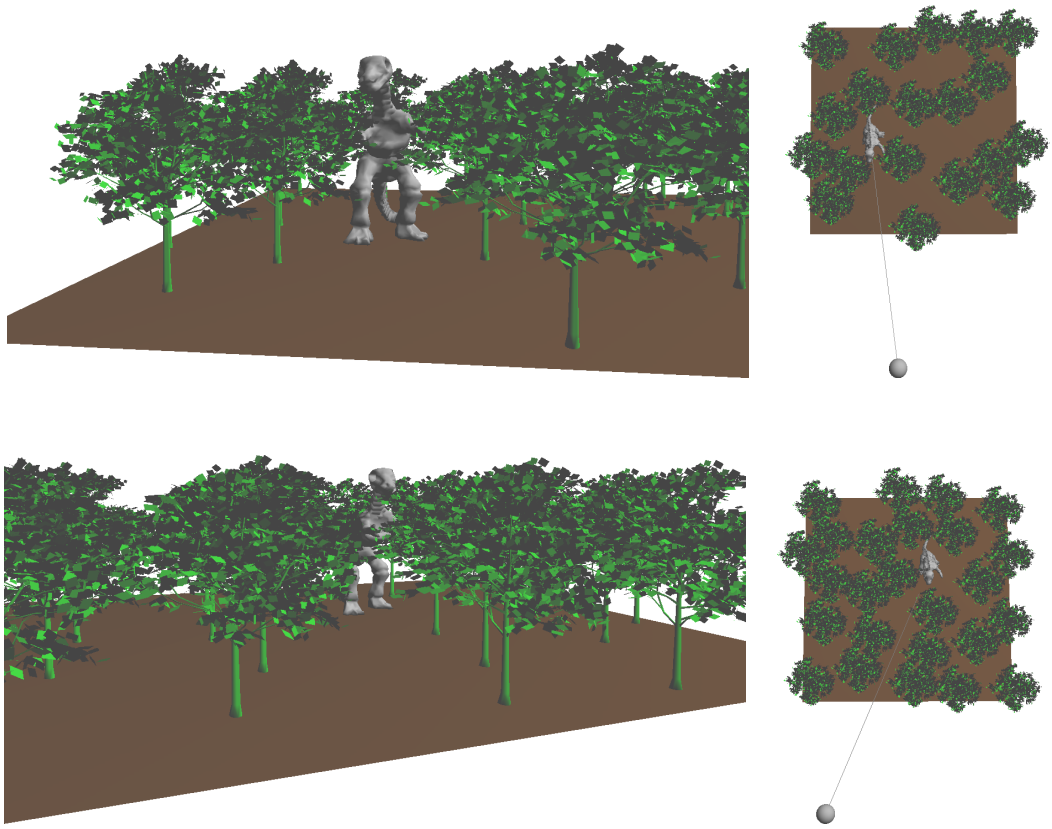

Figura 6.11: Resultados da seleção automática da melhor visão na cena da floresta: a máquina teve sucesso no posicionamento da câmera em outras florestas, gerando visões onde era possível visualizar o dinossauro.

do grupo. Usando as Galerias Inteligentes, o usuário deve navegar pelas galerias classificando visões como boas ou ruins e gerando novas galerias até chegar a uma visão final satisfatória. Ele pode também solicitar ajuda da máquina para selecionar automaticamente boas visões, e ordenar as galerias de acordo com a avaliação da máquina.

Nessa pré-avaliação simples, analisamos as curvas de aprendizagem e tempos dos usuários. As curvas de aprendizagem exibidas na figura 6.13 mostram que, após treinar dois ou três modelos, o tempo de interação do usuário com as Galerias Inteligentes diminui consideravelmente, enfatizando a capacidade de aprendizagem das Galerias Inteligentes.

Os tempos de interação estão diretamente relacionados com o acúmulo dos dados de treinamento, que possibilita a assistência das Galerias Inteligentes ao usuário, realizando seleções automáticas e ordenações nas galerias. Consequentemente, isto reduz o tempo que o usuário gasta para avaliar as visões de cada galeria. Por outro lado, os tempos de interação do usuário com o *Blender* permanecem praticamente constantes.

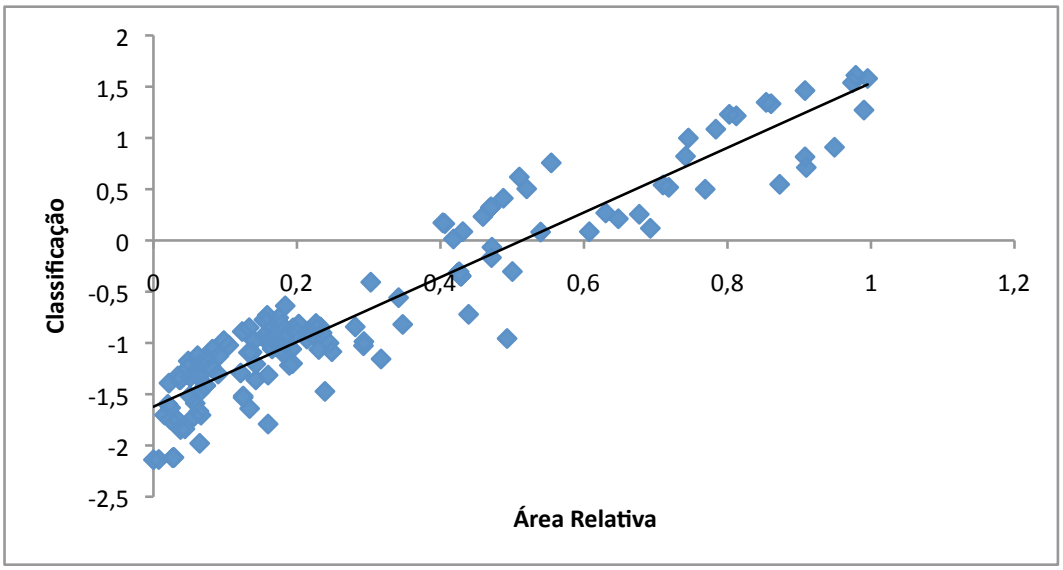

Figura 6.12: Regressão linear do gráfico do descritor de área relativa referente ao treino do dinossauro (figura 6.10): alta correlação linear.

6.3 Tempos de execução

A tabela 6.2 exibe dados de algumas das interações realizadas por usuários inexperientes com a interface das Galerias Inteligentes, incluindo o total de visões e galerias geradas, tempos de pré-processamento, processamento e de interação do usuário. Os tempos obtidos nas fases de processamento nos mostram que a interação com as Galerias Inteligentes foi realizada em tempo *quasi*-real.

Como boa parte do processamento intrínseco dos modelos é realizada em fase de pré-processamento, sua complexidade não afeta consideravelmente os tempos de processamento durante a interação do usuário. O cálculo da função de classificação do SVM e a avaliação de parâmetros também têm custo computacional desprezível, visto que os bancos de dados compostos por parâmetros avaliados por usuários são da ordem de algumas dezenas ou centenas de parâmetros. Consequentemente, o maior gargalo das Galerias Inteligentes está na fase de extração de descritores, mais especificamente na avaliação de cada imagem, por cada descritor. Neste caso, a resolução da imagem influencia fortemente o tempo de execução, visto que boa parte dos descritores deve varrer todos os pixels da imagem.

Além disso, descritores mais elaborados, como o que mede a complexidade da silhueta, podem ser adaptados para processamento em GPU. Em particular, podemos avaliar a complexidade da silhueta no espaço do objeto, extraindo sua silhueta 3D com tempo de execução menor.

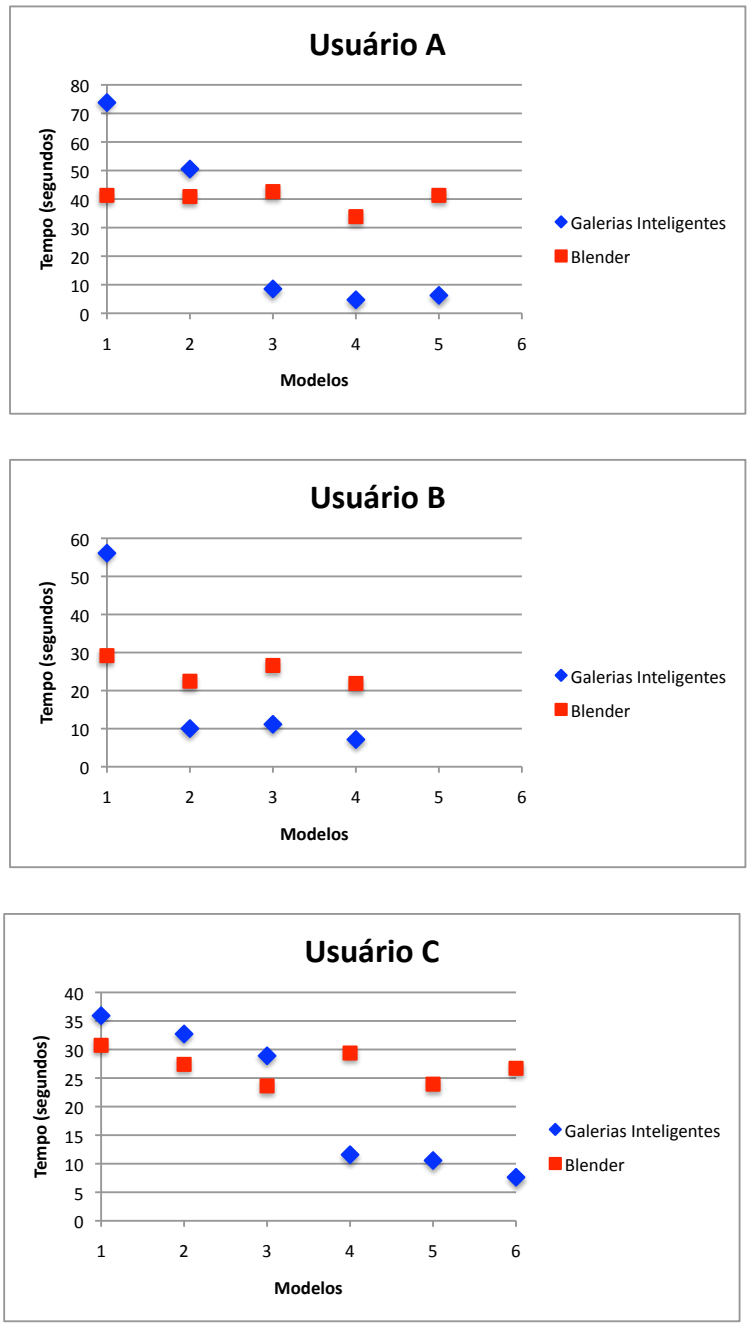

Figura 6.13: Curvas de aprendizagem de três usuários: o tempo de interação com a interface das Galerias Inteligentes diminui com o tempo, em contraste com interfaces convencionais, onde o tempo de interação permanece constante.

Finalmente, considerando que os tempos obtidos foram consequência de um processamento sequencial de parâmetros e descritores, estimamos que para atingirmos tempos interativos, pode ser suficiente paralelizar a avaliação dos descritores e dos parâmetros.

Modelo	**# de visões**	**# de galerias**	**Pré-Proc. seg.**	**Proc. seg.**	**Interação seg.**
vaca (0)	60	4	2.140	8.280	73.8
cão (1)	44	2	4.218	6.595	50.
gato (2)	20	0	8.875	3.281	8.5
dinossauro (3)	20	0	17.172	3.469	4.7
pantera (4)	20	0	4.703	3.234	6.3
nó 3.1c (0)	36	2	6.890	4.671	56.1
nó 7.7c (1)	20	0	6.672	3.235	10.0
nó 6.13c (2)	20	0	3.375	3.250	11.1
nó 138k (3)	20	0	3.735	3.250	7.1
triceratops (0)	28	1	2.313	4.906	35.9
dragão (1)	24	1	21.328	3.250	32.7
polvo (2)	24	1	9.531	4.375	28.9
mosquito (3)	20	0	5.141	3.234	11.6
mulher (4)	20	0	4.359	3.297	10.6
esquilo (5)	20	0	5.281	3.328	7.6

Tabela 6.2: Eficiência da interface das Galerias Inteligentes. As linhas horizontais dividem treinos de diferentes usuários.

7
Limitações

Discutimos neste capítulo as duas principais limitações de nosso método.

7.1
Dimensão do espaço de parâmetros

Em nosso método, o espaço de parâmetros Ω é parametrizado em um hipercubo $\mathcal{C}$ de dimensão n, correspondente à dimensão de Ω. Esta parametrização é necessária para garantir a consistência das combinações convexas, que são realizadas no hipercubo (ver apêndice B).

Para garantir que um conjunto finito de parâmetros Ψ tenha potencial para cobrir o conjunto Ω realizando apenas combinações convexas, é necessário garantir que pelo menos os parâmetros correspondentes a cada vértice do hipercubo $\mathcal{C}$ pertença a Ψ. Isto implica que precisamos de pelo menos 2^n parâmetros em Ψ, e consequentemente pelo menos 2^n parâmetros na galeria inicial. Porém, como as combinações convexas são aleatórias, é recomendável um número maior de amostras em Ψ para garantir o bom funcionamento das Galerias Inteligentes.

Vimos no capítulo 4 que o excesso de parâmetros nas galerias torna o trabalho do usuário exaustivo e a interface incompreensível. Para manter a quantidade de parâmetros por galeria menor ou igual a 36 parâmetros, é necessário restringir a dimensão n de Ω para 6 no máximo. Em diversos problemas de ajuste de parâmetros, esta quantidade é suficiente.

No problema de posicionamento de câmera, por exemplo, esta dimensão chega a 6 se deixarmos livres apenas a posição da câmera, direção de visão e orientação da câmera. Porém, neste caso extremo não conseguimos bons resultados, pois frequentemente o usuário não conseguia encontrar um parâmetro satisfatório explorando a interface das galerias.

Uma alternativa para problemas de dimensão mais alta é a geração de mais de uma galeria inicial, onde os parâmetros seriam divididos em várias galerias, e o usuário teria um trabalho mais exaustivo de treino e seleção de

parâmetros. Este seria um cenário onde o auxílio da máquina na seleção e ordenação de parâmetros pode reduzir drasticamente o esforço do usuário.

Uma segunda alternativa, a qual adotamos em boa parte de nossos experimentos com o problema de posicionamento de câmera, é restringir o espaço de parâmetros Ω em algumas dimensões. Por exemplo, para visualizar modelos simples, como animais, restringimos a posição da câmera a uma esfera centrada no centro do objeto, com direção de visão fixada no centro do objeto. Esta restrição diminuiu a dimensão de nosso espaço de parâmetros Ω para 3, aumentando a satisfação do usuário com o resultado final.

Uma terceira alternativa é usar aprendizagem semi-supervisionada.

7.2 Restrição a instâncias similares do problema

As Galerias Inteligentes consideram que os dados de treino são obtidos de instâncias similares de um dado problema, e devem ser aplicadas em instâncias similares deste problema. A definição de instâncias similares neste caso não é bem posta, e está diretamente relacionada à semântica do conjunto de descritores atribuídos ao problema.

Por exemplo, no problema de posicionamento de câmera, não se espera que o treino realizado em modelos como rostos humanos seja aplicável a nós tridimensionais, como ilustra a figura 7.1. Neste caso, o treino dos rostos não obteve resultados significativos quando aplicado para posicionamento automático de câmera no nó, e vice-versa.

Portanto, deve estar claro para o usuário não só a classe de instâncias na qual ele vai trabalhar, mas também seu objetivo comum como, por exemplo, visões laterais de animais.

Uma alternativa seria classificar o tipo de forma e escolher o dado de treino adequado, mas isso requer mais esforço em visão computacional 3D.

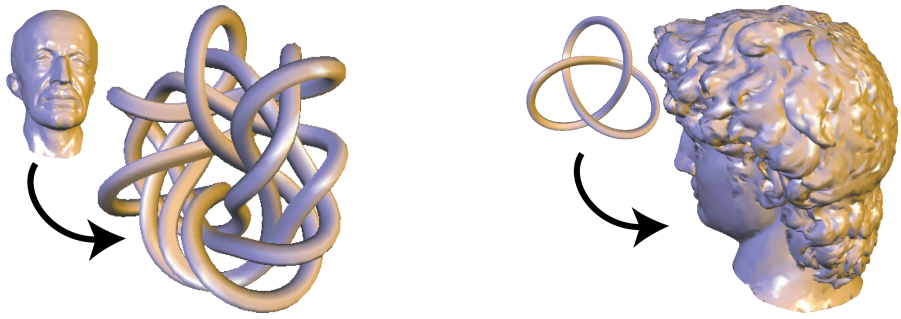

Figura 7.1: Restrição a instâncias similares: o treino do Max Planck e David não obtém resultados significativos quando aplicado ao nó (esquerda), enquanto o treino do nó também obtém um resultado sem significado no David.

8
Conclusão

Apresentamos as Galerias Inteligentes, uma nova abordagem para ajustar parâmetros em problemas subjetivos da Computação Visual. Nosso método estende as Galerias de Design realizando aprendizagem supervisionada das preferências dos usuários, que ensinam gradualmente uma máquina de aprendizagem enquanto navegam nas galerias para encontrar seus parâmetros ótimos.

Estudamos também o problema de posicionamento de câmera, analisando, escolhendo e desenvolvendo descritores que expressam características relevantes das visões, e que consequentemente influenciam o processo de classificação. Em contraste com outros métodos de posicionamento automático de câmera, as Galerias Inteligentes são capazes de selecionar automaticamente as melhores visões de acordo com as preferências de cada usuário.

Em experimentos realizados no problema de posicionamento de câmera, nosso método mostra-se capaz de assimilar e reproduzir as preferências do usuário em diversos cenários, desde modelos simples similares, como animais, até modelos desafiantes, como nós e cenas com grande potencial de oclusão. Nosso método também se mostra capaz de respeitar a subjetividade do problema, retornando diferentes resultados de acordo com preferências de diferentes usuários. Em comparações realizadas, conseguimos obter resultados robustos inclusive quando outros métodos recentes falham.

A robustez do método, unida com a satisfação do usuário e a possibilidade de interação em tempo *quasi*-real nos leva a concluir que nosso método de fato é uma boa opção para tratar problemas desse porte.

Dentre as limitações identificadas, segue em aberto o problema de geração de galerias contendo parâmetros de dimensão muito alta, onde nosso método requer no mínimo 2^n amostras na galeria. Este problema também tem relação direta com a organização de nossa interface. O potencial de dispor de uma máquina para classificar e ordenar parâmetros pode ser melhor aproveitado em termos de organização dos parâmetros na galeria, simplificando ainda mais o trabalho do usuário.

Apresentamos métodos para estimar a influência de cada descritor na

classificação de parâmetros, e para realizar calibração de parâmetros da máquina de aprendizagem, os quais nos possibilitam aumentar a robustez de nosso método. Porém, o método heurístico adotado para estimar a influência dos descritores não é capaz de detectar padrões não lineares nos gráficos g_l^{Γ}. O uso e desenvolvimento de métodos para realizar este tipo de análise não-linear pode aperfeiçoar ainda mais nosso método, além de prover uma ferramenta mais robusta para determinar a influência de parâmetros nos mais diversos problemas. Em particular, isto poderia ser aplicado na engenharia de descritores de outros problemas, para determinar quais descritores devem ser incluídos na solução.

Este trabalho abre um leque de possibilidades de problemas subjetivos de seleção de parâmetros que se adaptariam às Galerias Inteligentes. Opções de trabalhos futuros nesta linha incluem posicionamento de luz, *design* de funções de transferência e seleção de isosuperfícies.

Finalmente, exibimos resultados preliminares obtidos para posicionamento de câmera em uma cena de animação. Realizamos um treino em apenas dois quadros chave (*keyframe*) (ver figura 8.1), e depois solicitamos ajuda das Galerias Inteligentes para selecionar, a cada 2 segundos, as 5 melhores posições de câmera do respectivo quadro chave. Para evitar oscilações da câmera, selecionamos, para cada quadro chave, a posição de câmera que de modo global minimizasse o deslocamento das imagens. As visões intermediárias foram interpoladas usando *splines* cúbicas. O resultado é exibido na figura 8.2 e sua qualidade nos motiva a continuar este trabalho.

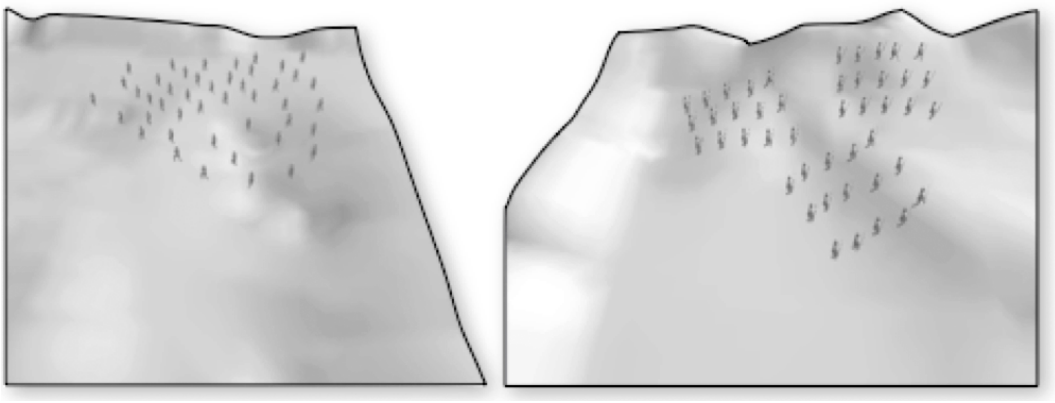

Figura 8.1: Quadros treinados da cena de animação. As visões que focam nos grupos são selecionadas como boas.

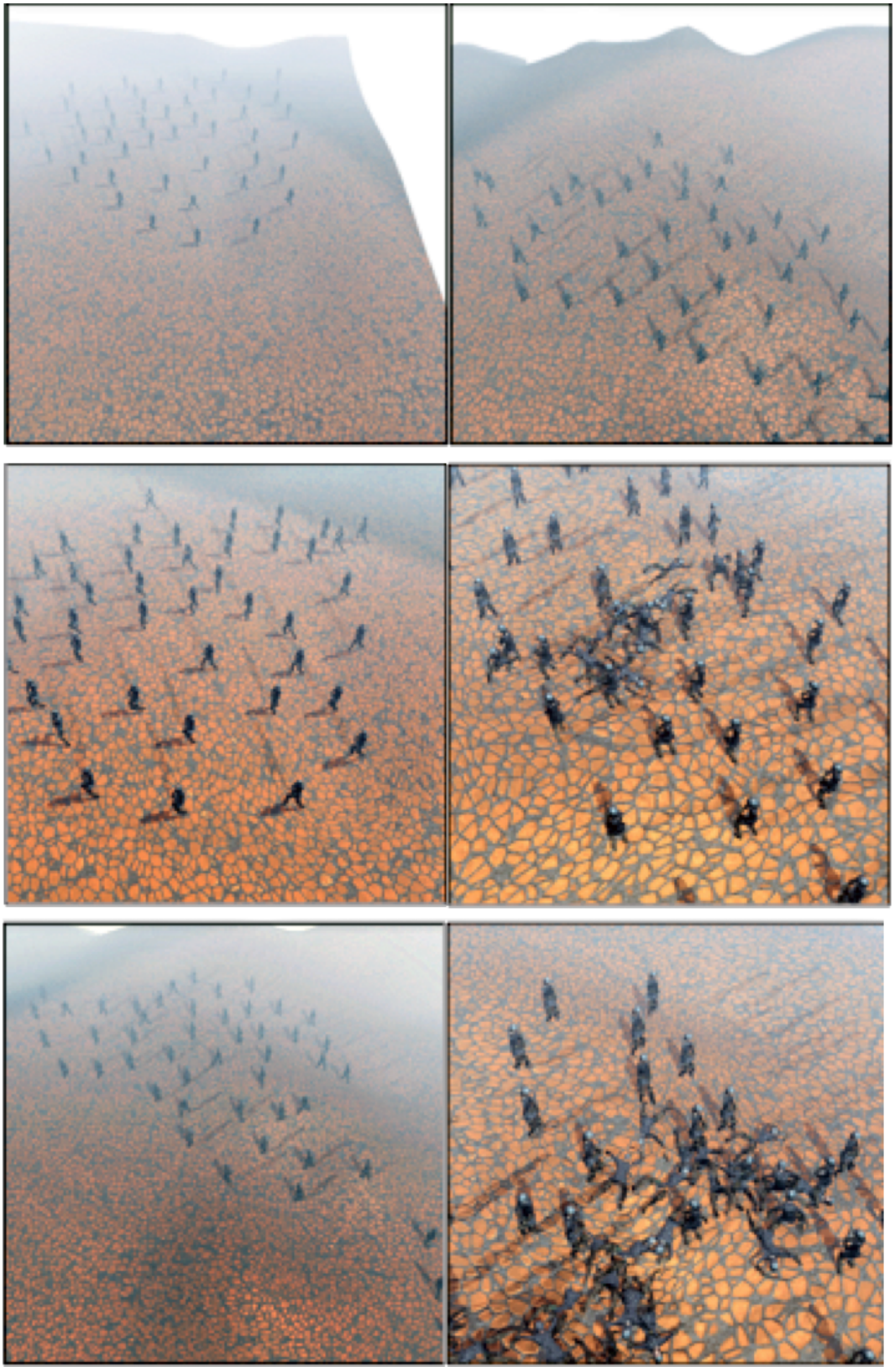

Figura 8.2: Visões selecionadas automaticamente nos outros quadros chave da animação: a máquina é capaz de selecionar visões relevantes da animação.

Referências Bibliográficas

[1] BANZHAF, W.; NORDIN, P.; KELLER, R. ; FRANCONE, F.. **Genetic Programming - An Introduction**. Morgan Kaufmann, 1998.

[2] BARES, W.; MCDERMOTT, S.; BOUDREAUX, C. ; THAINIMIT, S.. **Virtual 3D camera composition from frame constraints**. In: MULTIMEDIA, p. 177–186. ACM, 2000.

[3] BARES, W. H.. **A Photographic Composition Assistant for Intelligent Virtual 3D Camera Systems**. In: SMART GRAPHICS, p. 172–183, 2006.

[4] BLANZ, V.; TARR, M. J.; BÜLTHOFF, H. H. ; VETTER, T.. **What object attributes determine canonical views**. Perception, 28(5):575–600, 1999.

[5] BORDIGNON, A.; PEREIRA, R.; LOPES, H.; LEWINER, T. ; TAVARES, G.. **Exploratory visualization based on multidimensional transfer functions and star coordinates**. In: SIBGRAPI 2006 (XIX BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), p. 273–280, Manaus, AM, october 2006. IEEE.

[6] BORDIGNON, A.; VATH, B.; VIEIRA, T.; FERREIRA, C.; CRAIZER, M. ; LEWINER, T.. **Scale-space for union of 3d balls**. In: SIBGRAPI 2009 (XXII BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), Rio de Janeiro, RJ, october 2009. IEEE.

[7] BOSER, B. E.; GUYON, I. M. ; VAPNIK, V. N.. **A Training Algorithm for Optimal Margin Classifiers**. In: COMPUTATIONAL LEARNING THEORY, p. 144–152. ACM, 1992.

[8] CHANG, C.-C.; LIN, C.-J.. **LIBSVM: a library for support vector machines**, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[9] CHRISTIE, M.; MACHAP, R.; NORMAND, J.-M.; OLIVIER, P. ; PICKERING, J.. **Virtual Camera Planning: A Survey**. In: SMART GRAPHICS, p. 40–52, 2005.

[10] CHRISTIE, M.; OLIVIER, P. ; NORMAND, J.-M.. **Camera Control in Computer Graphics**. Computer Graphics Forum, 27(7), 2008.

[11] CORTES, C.; VAPNIK, V.. **Support-vector networks**. Springer, 1995.

[12] DRUCKER, S. M.; ZELTZER, D.. **Intelligent Camera Control in a Virtual Environment**. In: GRAPHICS INTERFACE, p. 190–199, 1994.

[13] FEIJO, B.; PAGLIOSA, P. ; CLUA, E. W. G.. **Visualização, simulação e games**. In: XXIV JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA DO CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, volumen 1, p. 151–223, Rio de Janeiro, RJ, 2006. Editora PUC-Rio.

[14] GÓMEZ, F.; HURTADO, F.; SELLARÈS, J. A. ; TOUSSAINT, G.. **Nice Perspective Projections**. Visual Communication and Image Representation, 12(4):387–400, 2001.

[15] GOOCH, B.; REINHARD, E.; MOULDING, C. ; SHIRLEY, P.. **Artistic Composition for Image Creation**. In: RENDERING TECHNIQUES, p. 83–88. Springer, 2001.

[16] HALPER, N.; OLIVER, P.. **CamPlan: A Camera Planning Agent**. In: SMART GRAPHICS, p. 92–100, 2000.

[17] HE, L.-W.; COHEN, M. F. ; SALESIN, D. H.. **The virtual cinematographer: a paradigm for automatic real-time camera control and directing**. In: SIGGRAPH, p. 217–224. ACM, 1996.

[18] HE, T.; HONG, L.; KAUFMAN, A. ; PFISTER, H.. **Generation of transfer functions with stochastic search techniques**. In: VISUALIZATION, p. 227–234, Los Alamitos, CA, USA, 1996. IEEE.

[19] JANKUN-KELLY, T. J.; MA, K.-L.. **Visualization exploration and encapsulation via a spreadsheet-like interface**. IEEE Transactions on Visualization and Computer Graphics, 7(3):275–287, 2001.

[20] KAMADA, T.; KAWAI, S.. **A simple method for computing general position in displaying three-dimensional objects**. Computer Vision, Graphics, Image Processing, 41(1):43–56, 1988.

[21] LAGA, H.; NAKAJIMA, M.. **Supervised Learning of Salient 2D Views of 3D Models**. In: NICOGRAPH, 2007.

[22] LAGE, M.; LEWINER, T.; LOPES, H. ; VELHO, L.. **Chf: a scalable topological data structure for tetrahedral meshes**. In: SIBGRAPI 2005 (XVIII BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), Natal, RN, october 2005. IEEE.

[23] LEE, C. H.; VARSHNEY, A. ; JACOBS, D. W.. **Mesh saliency**. In: SIGGRAPH, p. 659–666. ACM, 2005.

[24] LEWINER, T.; AO D. GOMES, J.; LOPES, H. ; CRAIZER, M.. **Arc-length based curvature estimator**. In: SIBGRAPI 2004 (XVII BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), p. 250–257, Curitiba, PR, october 2004. IEEE.

[25] LIM, I. S.; DE HERAS CIECHOMSKI, P.; SARNI, S. ; THALMANN, D.. **Planar arrangement of high-dimensional biomedical data sets by Isomap coordinates**. In: IEEE SYMPOSIUM ON COMPUTER-BASED MEDICAL SYSTEMS. IEEE, 2003.

[26] MARKS, J.; ANDALMAN, B.; BEARDSLEY, P. A.; FREEMAN, W.; GIBSON, S.; HODGINS, J.; KANG, T.; MIRTICH, B.; PFISTER, H.; RUML, W.; RYALL, K.; SEIMS, J. ; SHIEBER, S. M.. **Design galleries: a general approach to setting parameters for computer graphics and animation**. In: SIGGRAPH, p. 389–400. ACM, 1997.

[27] MAXIMO, A.; MARROQUIM, R.; ESPERANÇA, C.; DOS SANTOS, R. W.; BENTES, C. ; FARIAS, R.. **High-performance volume rendering for 3d heart visualization**. In: WORKSHOP ON HIGH PERFORMANCE COMPUTING IN THE LIFE SCIENCES - HPCLIFE 2006, OURO PRETO-MG, October 2006.

[28] MEYER, M.; DESBRUN, M.; SCHRÖDER, P. ; BARR, A.. **Discrete differential–geometry operators for triangulated 2–manifolds**. In: VISMATH, p. 35–57, 2002.

[29] NGAN, A.; DURAND, F. ; MATUSIK, W.. **Image-driven navigation of analytical brdf models**. In: RENDERING TECHNIQUES, p. 399–408, June 2006.

[30] PASSOS, E.; JOSELLI, M.; ZAMITH, M.; ROCHA, J.; MONTENEGRO, A. A.; CONCI, A.; FEIJO, B. ; CLUA, E. W. G.. **Supermassive crowd**

simulation on gpu based on emergent behavior. CIE, 7(2):145–155, 2009.

[31] PINTO, F.; FREITAS, C.. **Two-level interaction transfer function design combining boundary emphasis, manual specification and evolutive generation**. In: SIBGRAPI 2006 (XIX BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), p. 281–288, Manaus, AM, october 2006. IEEE.

[32] PLATT, J. C.. **Fast training of support vector machines using sequential minimal optimization**. MIT Press, Cambridge, MA, USA, 1999.

[33] PLEMENOS, D.; BENAYADA, M.. **Intelligent Display in Scene Modeling: New Techniques to Automatically Compute Good Views**. In: GRAPHICON, 1996.

[34] POLONSKY, O.; PATANÈ, G.; BIASOTTI, S.; GOTSMAN, C. ; SPAGNUOLO, M.. **What's in an image?** The Visual Computer, 21(8–10):840–847, 2005.

[35] RICARDO MARROQUIM; ANDRE MAXIMO; RICARDO FARIAS ; CLAUDIO ESPERANÇA. **Volume and Isosurface Rendering with Gpu-accelerated Cell Projection**. Computer Graphics Forum, 27:24–35, 2008.

[36] SCHAREIN, R. G.. **Interactive Topological Drawing**. PhD thesis, Department of Computer Science, The University of British Columbia, 1998.

[37] SCHÖLKOPF, B.; SMOLA, A. J. ; MÜLLER, K.-R.. **Kernel principal component analysis**, p. 327–352. MIT, 1999.

[38] SHAMIR, A.. **A survey on mesh segmentation techniques**. Computer Graphics Forum, 27(6):1539–1556, 2008.

[39] SHAPIRA, L.; SHAMIR, A. ; COHEN-OR, D.. **Image Appearance Exploration by Model-Based Navigation.** Computer Graphics Forum, 28(2):629–638, 2009.

[40] SHILANE, P.; FUNKHOUSER, T.. **Distinctive regions of 3D surfaces**. Transactions on Graphics, 26(2):7, 2008.

[41] STOEV, S.; STRASSER, W.. **A Case Study on Automatic Camera Placement and Motion for Visualizing Historical Data**. In: VISUALIZATION. IEEE, 2002.

[42] SYSTEMS, A.. **After effects cs4**. Software, Outubro 2008.

[43] SYSTEMS, A.. **Photoshop cs4**. Software, Outubro 2008.

[44] TARJAN, R. E.. **Efficiency of a good but not linear set union algorithm**. Journal of the ACM, 22(2):215–225, 1975.

[45] VAPNIK, V.. **The Nature of Statistical Learning Theory**. Springer, 2000.

[46] VÁZQUEZ, P.-P.; FEIXAS, M.; SBERT, M. ; HEIDRICH, W.. **Viewpoint Selection using Viewpoint Entropy**. In: VISION MODELING AND VISUALIZATION, p. 273–280. Aka, 2001.

[47] VIEIRA, T.. **Registro Automático de Superfícies usando Spin-Images**. Master's thesis, Universidade Federal de Alagoas, Feb. 2007. Advised by Adelailson Peixoto.

[48] VIEIRA, T.; BORDIGNON, A.; LEWINER, T. ; VELHO, L.. **Geometry super-resolution by example**. In: SIBGRAPI 2009 (XXII BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING), Rio de Janeiro, RJ, october 2009. IEEE.

[49] VIEIRA, T.; BORDIGNON, A.; PEIXOTO, A.; TAVARES, G.; LOPES, H.; VELHO, L. ; LEWINER, T.. **Learning good views through intelligent galleries**. Computer Graphics Forum, 28:717–726, 2009.

[50] VIEIRA, T.; PEIXOTO, A.; VELHO, L. ; LEWINER, T.. **An iterative framework for registration with reconstruction**. In: VISION, MODELING, AND VISUALIZATION 2007, p. 101–108, Saarbrücken, november 2007. Pirrot.

A
Máquinas de Suporte Vetorial

A área de aprendizagem de máquina se preocupa com o desenvolvimento de algoritmos que permitam aos computadores aprender com dados de entrada provenientes, por exemplo, de sensores ou bancos de dados. O conhecimento adquirido pela máquina é utilizado para reconhecer automaticamente padrões complexos e tomar decisões inteligentes.

Uma das classes de algoritmos que tenta realizar aprendizagem de máquina é chamada de aprendizagem supervisionada. Esses algoritmos tentam deduzir uma função a partir de dados de treino contendo a resposta esperada da máquina. Em geral, esses dados de treino consistem de objetos (*input*), normalmente na forma de vetores, associados a saídas (*output*) desejadas. Dependendo do problema, essas saídas podem ser representadas por um valor contínuo, caracterizando um problema de regressão, ou valores discretos, também chamados de classes, caracterizando um problema de classificação.

As Máquinas de Suporte Vetorial são um conjunto de métodos de aprendizagem supervisionada usadas para classificação e regressão de dados. Em geral, esses métodos calculam um hiperplano, ou conjunto de hiperplanos, cujos parâmetros serão utilizados na função de classificação ou regressão. Vamos nos focar agora no problema de classificação binária. Para maiores detalhes, referenciamos o leitor ao livro de Vapnik (45).

A.1
Classificador linear

Inicialmente, a máquina recebe um conjunto de dados de treino (*training set*) na forma:

$$\Gamma = \{(x_i, c_i) | x_i \in \mathbb{R}^p, c_i \in \{-1, 1\}\}_{i=1}^{N}, \tag{A-1}$$

onde x_i representa um dado de entrada p-dimensional, e c_i sua classificação.

Nosso objetivo é encontrar um hiperplano que separe, da melhor maneira possível, os pontos com classificação $c_i = -1$ dos pontos classificados com

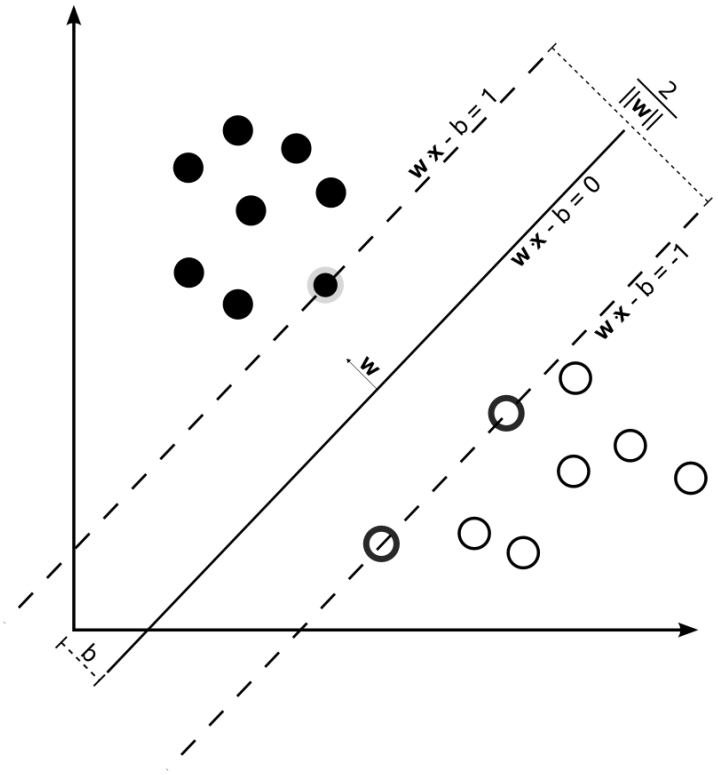

Figura A.1: Exemplo de hiperplano de margem máxima separando duas classes.

$c_i = 1$, como ilustrado na figura A.1. Este hiperplano será chamado de hiperplano de margem máxima.

Todo hiperplano pode ser descrito por uma equação do tipo

$$w \cdot x - b = 0, \tag{A-2}$$

onde $w \neq 0$ é o vetor normal do hiperplano e $\frac{b}{\|w\|}$ determina o deslocamento do hiperplano a partir da origem, ao longo do vetor w.

Queremos escolher w_{opt} e b_{opt} de modo que os hiperplanos paralelos dados pelas equações

$$w_{opt} \cdot x - b_{opt} = 1$$

e

$$w_{opt} \cdot x - b_{opt} = -1$$

maximizem a distância entre si, dada por $\frac{2}{\|w\|}$, respeitando a condição de separar os dados de acordo com suas classes. O hiperplano de margem máxima será dado pela equação $w_{opt} \cdot x - b_{opt} = 0$.

Estabelecemos, portanto, um problema de otimização com restrições, onde queremos maximizar a distância entre os planos $\frac{2}{\|w\|}$, considerando restrições que nos garantam que esses hiperplanos separem os dados de acordo com sua classe, ou seja:

$$c_i(w \cdot x_i - b) \geq 1, \quad i = 1 \ldots N. \tag{A-3}$$

Com uma pequena adaptação nesta formulação, chegamos à forma primal, caracterizada como um problema de otimização quadrática, que explicitamos abaixo:

Minimize (em w, b)

$$\frac{1}{2}\|w\|^2$$

sujeito a

$$c_i(w \cdot x_i - b) \geq 1, \quad i = 1 \ldots N.$$

Para deduzir a forma dual Lagrangeana, usamos a função Lagrangeana generalizada:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left[c_i(w \cdot x_i + b) - 1\right], \tag{A-4}$$

onde α_i são os multiplicadores de Lagrange.

Minimizando $L(w, b, \alpha)$ em relação a w e b, chegamos às relações

$$w = \sum_{i=1}^{N} \alpha_i c_i x_i \tag{A-5}$$

e

$$\sum_{i=1}^{N} \alpha_i c_i = 0.$$

Substituindo estas relações de volta em $L(w, b, \alpha)$ chegamos à forma dual:

Maximize (em α_i)

$$\mathcal{W}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j c_i c_j x_i^T x_j \tag{A-6}$$

sujeito a

$$\alpha_i \geq 0, \quad i = 1 \ldots N \tag{A-7}$$

e

$$\sum_{i=1}^{n} \alpha_i c_i = 0, \quad i = 1 \ldots N. \tag{A-8}$$

Usando esta formulação dual, é possível encontrar w_{opt} substituindo-se os valores de α_{opt} na equação (A-5). Para encontrar b_{opt}, é necessário voltar à forma primal:

$$b_{opt} = -\frac{\max_{c_i=-1}(w_{opt} \cdot x_i) + \min_{c_i=1}(w_{opt} \cdot x_i)}{2}.$$

De acordo com a condição de complementaridade de Karush-Kuhn-Tucker, as soluções ótimas $\alpha_{opt}, w_{opt}, b_{opt}$ devem satisfazer

$$(\alpha_i)_{opt}\,[c_i(w_{opt}\cdot x_i + b) - 1] = 0, \quad i = 1 \ldots N.$$

Isto significa que um ponto x_i só pode ter seu multiplicador de Lagrange α_i não nulo se $c_i(w_{opt}\cdot x_i + b) = 1$, ou seja, quando este ponto pertence a um dos hiperplanos que define a margem. Este resultado é extremamente importante, pois, de acordo com a equação (A-5), só precisamos dos pontos x_i na margem para representar w. Esses pontos serão chamados de vetores de suporte. É interessante enfatizar que os vetores de suporte são geralmente poucos em relação ao tamanho do training set, pois só ocorrem na fronteira entre as classes.

Uma vez identificados os vetores de suporte x_i com seus respectivos α_i, podemos expressar nossa função classificadora como:

$$\hat{f}(x) = \sum_{i \in SV} [\alpha_i c_i (x_i \cdot x)] + b, \tag{A-9}$$

onde SV é o conjunto com os índices dos vetores de suporte.

Quando $\hat{f}(x) = 0$, o ponto x está contido no hiperplano que divide as amostras. Em qualquer outro caso, usamos $sign(\hat{f}(x))$ para classificar um ponto x como pertencente à classe 1 ou -1.

Até agora, assumimos que os dados são linearmente separáveis, ou seja, que é possível separar dados de diferentes classes usando apenas um hiperplano. Descreveremos agora duas generalizações deste método para tratar dados ruidosos e dados não linearmente separáveis.

A.2
Classificador linear com margem flexível

Em muitos cenários é necessário trabalhar com conjuntos de dados de treino que não são linearmente separáveis por causa de ruído nas classificações, por exemplo. Cortes *et al.* (11) sugeriram uma modificação da formulação original para tratar este tipo de problema. A idéia é que, quando não for possível encontrar um plano que divida os pontos das duas classes, pode ser suficiente encontrar o hiperplano que minimiza o erro de classificação dos pontos de treino.

Para medir o possível erro de classificação do hiperplano, são introduzidas variáveis ξ_i associadas às restrições da forma primal do problema original (figura A.2):

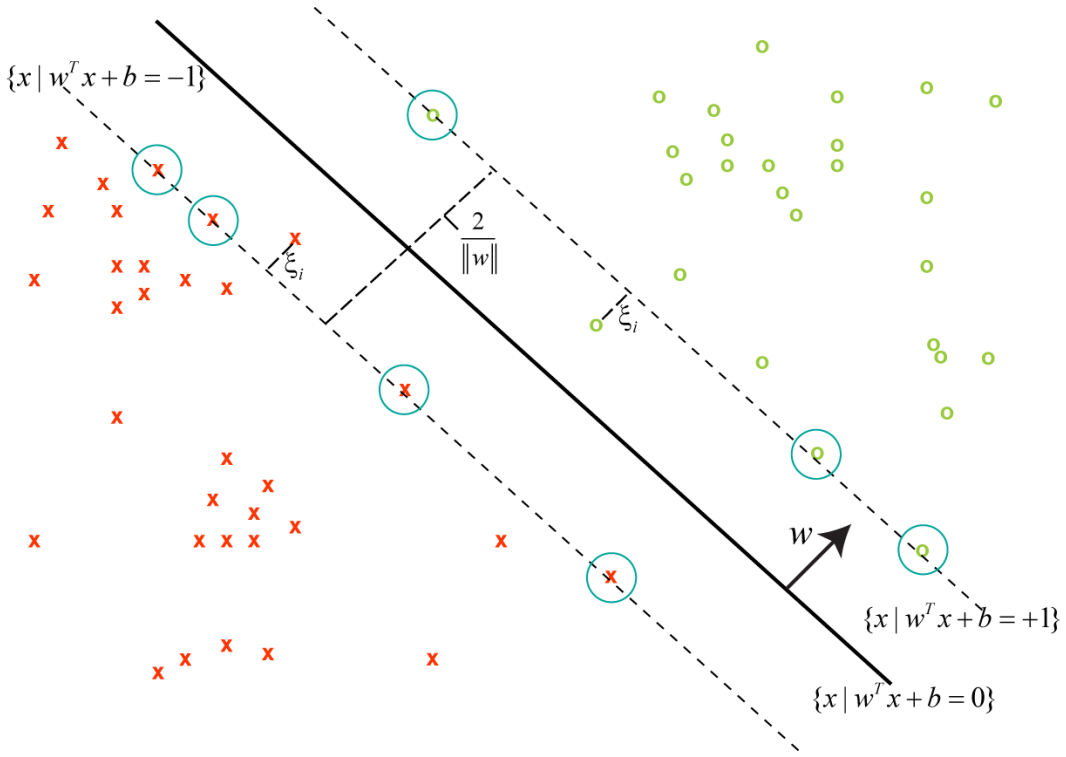

Figura A.2: Formulação das Máquinas de Suporte Vetorial usando margem flexível: a condição do hiperplano ótimo é relaxada, permitindo que algumas amostras ultrapassem a margem.

$$c_i(w \cdot x_i - b) \geq 1 - \xi_i, \quad i = 1 \ldots n. \tag{A-10}$$

Para minimizar este erro, introduzimos um novo termo na função objetivo da minimização:

$$\frac{1}{2}\|w\|^2 + C\sum_i \xi_i, \tag{A-11}$$

onde C é um peso associado ao erro tolerável. A solução deste problema de minimização é similar ao procedimento descrito na seção anterior.

A.3
Classificador não-linear

Quando os dados de treino não forem linearmente separáveis, é possível recorrer a uma extensão proposta por Boser *et al.* (7). Neste método, a função de classificação $\hat{f}$ corresponde a uma função linear em um espaço de Hilbert $\mathcal{F}$, chamado de *feature space*. Em particular, o hiperplano deduzido para criação de $\hat{f}$ estará contido não mais no espaço de entrada $\mathbb{R}^p$, mas em um espaço de alta dimensão ou dimensão infinita $\mathcal{F}$, o que na prática significa que ele pode não ser linear no espaço de entrada.

Inicialmente, é necessário definir uma função não-linear $\phi: \mathbb{R}^p \mapsto \mathcal{F}$, chamada de *feature map*, que mapeia os pontos de treino $x_i \in \mathbb{R}^p$ em $\mathcal{F}$.

Dada uma função ϕ, definimos um *kernel* como uma função $K: \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$, tal que $\forall x_i, x_j \in \mathbb{R}^p$

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j).$$

Uma função do tipo *kernel* generaliza o produto interno usual no espaço de entrada $\mathbb{R}^p$, e uma de suas vantagens é que ele possibilita mapear dados implicitamente para o *feature space* e treinar a máquina, sem a necessidade de conhecer uma função ϕ potencialmente complicada e problemática computacionalmente. A existência de ϕ a partir de K é garantida pelo Teorema de Mercer, se K for simétrica positiva definida. Um exemplo bastante utilizado é o *kernel* Gaussiano, dado por

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right). \tag{A-12}$$

Neste caso, $\mathcal{F}$ é um espaço de dimensão infinita, mas K é simples, pois é multiplicativo em dimensão.

Na prática, obtemos um classificador não-linear usando o mesmo algoritmo descrito na seção A.1, apenas substituindo cada produto interno pelo *kernel* não linear escolhido. A função classificadora resultante será dada por

$$\hat{f}(x) = \sum_{i \in SV} [\alpha_i c_i K(x_i, x)] + b. \tag{A-13}$$

É interessante notar que em momento algum é necessário conhecer a função ϕ.

A.4
Adequação às Galerias Inteligentes

A forma dual das Máquinas de Suporte Vetorial caracteriza um problema de otimização quadrática convexa (equação (A-6)) com restrições lineares (equações (A-7) e (A-8)). Como neste tipo de problema a solução é única e não existem máximos locais, é possível encontrar esta solução eficientemente com custo proporcional à quantidade de dados de entrada N. Um método eficiente e paralelizável para este propósito é o SMO (32). A LibSVM (8) é uma boa alternativa de biblioteca de código aberto em C++ que implementa um método do tipo SMO.

Como mencionado anteriormente, a quantidade de vetores de suporte M é em geral bem menor que a quantidade de amostras N. Em particular, isso torna o custo de avaliar um dado de entrada x_i dado pela equação (A-9) muito

eficiente, com ordem de grandeza $O(M) \ll O(N)$.

Além disso, os problema de otimização e de avaliação dos dados de entrada não dependem diretamente da dimensão dos dados de entrada p, devido ao uso de *kernels*. Apenas a avaliação do *kernel* nos pontos é dependente da dimensão p. Isto implica que é possível acrescentar descritores sem prejudicar o custo computacional.

Finalmente, em termos de memória, uma máquina é representada apenas pelos vetores de suporte, além dos vetores w e b do hiperplano e parâmetros do *kernel* K, o que garante baixo custo de armazenamento.

B
Algoritmos Genéticos

Os algoritmos genéticos representam uma classe de algoritmos que tentam achar soluções aproximadas em problemas de otimização. Esses algoritmos recebem este nome porque se baseiam em princípios da biologia evolucionária como herança, mutação, seleção e *crossover*. Descreveremos a seguir os passos de um algoritmo genético, e o algoritmo usado neste trabalho no contexto de otimização de parâmetros. Recomendamos ao leitor o livro de Banzhaf *et al.* (1).

B.1
Algoritmo genético usual

Um algoritmo genético tem como objetivo encontrar, em uma população, o indivíduo que tem a melhor avaliação de acordo com uma função objetivo $\hat{f}: \Omega \mapsto \mathbb{R}$. Esta busca segue por diversas gerações de populações, seguindo o princípio de que a população evolui a cada nova geração, até que algum critério de parada seja atingido.

A elaboração de um algoritmo genético para resolver um problema específico requer:

1. estabelecer uma representação genética dos indivíduos, ou seja, definir o domínio Ω que representa os genes do indivíduo;
2. criar a função objetivo $\hat{f}$.

Uma vez definidos o domínio Ω e a função objetivo $\hat{f}$, um algoritmo genético deve seguir os passos descritos no algoritmo B.1. Descreveremos a seguir cada etapa do algoritmo.

B.1.1
Inicialização

Nesta etapa, é necessário estabelecer algum procedimento para gerar uma população inicial $\Psi \subset \Omega$, que será dada como entrada para o algoritmo. Uma estratégia comum é amostrar uniformemente o domínio Ω.

entrada: População inicial $\Psi \subset \Omega$
saída : Melhor indivíduo $\omega \in \Omega$
enquanto *nenhuma condição de parada atingida* **faça**
  Pais := Seleção(Ψ,$\hat{f}$);
  Ψ := Reprodução(Pais);
Retorne melhor parâmetro $\omega \in \Psi$ de acordo com $\hat{f}$.

Algoritmo B.1: Esqueleto de um algoritmo genético

B.1.2 Seleção

Em cada geração, indivíduos da população serão selecionados para reprodução. Para que a população evolua, devemos selecionar indivíduos portadores de bons genes, ou seja, indivíduos com boa avaliação de acordo com a função objetivo $\hat{f}$. Em geral, o processo de seleção ocorre como um processo estocástico, onde a probabilidade de um indivíduo ser selecionado depende de sua avaliação segundo $\hat{f}$. Isto tenta garantir que a qualidade média da população evolua com o tempo, visto que indivíduos com melhor avaliação tem maior probabilidade de se reproduzirem.

B.1.3 Reprodução

Nesta etapa, os pais selecionados na etapa anterior serão emparelhados para gerar uma nova geração de indivíduos. Cada par de pais pode passar por dois operadores genéticos: *crossover* e mutação. O primeiro operador recebe genes de dois indivíduos e gera um novo indivíduo, estabelecendo alguma estratégia para combinar genes dos pais. O segundo operador realiza pequenas mutações aleatórias no código genético, para garantir a variabilidade genética.

B.2 Algoritmo genético para otimização de parâmetros

Nesta seção descreveremos uma abordagem para resolver um problema de otimização de parâmetros usando um algoritmo genético baseado em combinações convexas.

Como descrito na seção anterior, precisamos inicialmente definir nossa representação genética dos indivíduos. Vamos considerar, como indivíduos, parâmetros $\omega \in \Omega \subset \mathbb{R}^n$, onde Ω é um hipercubo. Esta restrição é recomendável, visto que realizaremos combinações convexas entre elementos deste domínio. Entretanto, podemos usar esse hipercubo para representar domínios

de parâmetros mais complexos usando parametrizações. Deixaremos a função objetivo $\hat{f}$ em aberto.

Na fase de inicialização do algoritmo, é necessário criar um conjunto de amostras do domínio Ω. Para garantir que qualquer parâmetro $\omega \in \Omega$ possa ser gerado como resultado de combinações convexas do conjunto de amostras inicial, devemos incluir pelo menos os vértices do hipercubo Ω. Outras amostras podem ser geradas uniformemente no hipercubo.

Vamos considerar agora um conjunto de parâmetros $\Psi \subset \Omega$ selecionados do modo convencional para reprodução. Nesta fase, os parâmetros serão emparelhados aleatoriamente ν vezes, e cada par será reproduzido usando combinações convexas aleatórias. Desse modo, garantimos que cada pai vai gerar ν filhos, e a nova população será de $\frac{\nu}{2} \cdot |\Psi|$ indivíduos.

Seja (ω_i, ω_j) um dos pares criados aleatoriamente, e X uma variável aleatória uniforme em $[0, 1]$. Definimos nosso operador de *crossover* como $\xi\colon \Omega \times \Omega \mapsto \Omega$ dado por

$$\xi(\omega_i, \omega_j) = \omega_i + X(\omega_j - \omega_i).$$

Assim, o filho resultante tem probabilidade uniforme de estar em qualquer ponto da reta que liga os dois parâmetros no hipercubo.

Opcionalmente, podemos simular o efeito de uma mutação $\mathcal{M}\colon \Omega \mapsto \Omega$, perturbando aleatoriamente as coordenadas do novo indivíduo com uma pequena probabilidade. Na prática, este operador tem o poder de expandir a imagem $\xi(\Psi \times \Psi)$ para um subconjunto maior de Ω, de modo que $\xi(\Psi \times \Psi) \subset \mathcal{M}(\xi(\Psi \times \Psi))$, aumentando a variabilidade genética dos indivíduos.