Revista de Sistemas de Informação da FSMA
n. 28 (2021) pp. 35-44

SISTEMAS DE INFORMAÇÃO
http://www.fsma.edu.br/si/sistemas.html

TRILHA PRINCIPAL

Layout Semântico de Documentos com Interface Baseada em Agrupamento Hierárquico

Luis Carlos dos Santos Coutinho Retondaro, *M.Sc., CEFET/RJ*,
Claudio Esperança, *Ph.D., PESC/COPPE, Universidade Federal do Rio de Janeiro*

***Resumo*—A autoria de conteúdo multimídia é uma atividade comum em diversas áreas de aplicação como salas de aula, palestras, apresentações de negócios, reuniões, etc. Nesse processo, as tarefas fundamentais envolvem a organização espacial e temporal dos objetos. No entanto, preparar documentos que transmitam informação de qualidade e bem organizada é um processo que tem dois problemas desafiadores. Em primeiro lugar, o processo tradicional de agrupar objetos semanticamente relacionados tende a ser ineficiente, pois requer vários comandos individuais para realizar uma única operação. Em segundo lugar, a organização de hierarquias de elementos gráficos retirados de manuscritos ou fragmentos de imagens é normalmente realizada de forma pouco intuitiva. Neste artigo, propomos uma interface de manipulação direta na qual agrupamentos de objetos e seus níveis de organização são gerados automaticamente e sinalizados visualmente. Para avaliar a adequação dessas ideias a softwares de autoria, realizamos experimentos onde voluntários são instados a compor layouts simples usando tanto uma interface tradicional quanto uma baseada nas técnicas propostas. Resultados preliminares indicam que a interface que provê layout semântico automático conduz a uma melhor performance do usuário nessas tarefas.**

***Palavras-chave*—Layout de documentos, Clusterização hierárquica, Projeto de interação.**

Semantic Document Layout with Interface Based on Hierarchical Object Clustering

***Abstract*—**
The authoring of multimedia content is a common activity in several application areas such as classrooms, lectures, business presentations, meetings, etc. In this process, the fundamental tasks involve the spatial and temporal organization of objects. However, laying out documents that convey quality and well-organized information is a process that has two challenging problems. First, the traditional process of grouping semantically related objects together tends to be inefficient because it requires several individual commands to perform a single operation. Secondly, the organization of hierarchies of graphic elements taken from manuscripts or fragments of images is typically accomplished in an unintuitive way. In this article, we propose a direct manipulation interface in which object groupings and their organization levels are automatically generated and visually signaled. In order to assess the suitability of these ideas to authoring software, we carried out experiments where volunteers were asked to compose simple layouts using both a traditional interface and one based on the proposed techniques. Preliminary results indicate that the interface that provides automatic semantic layout leads to better user performance in these tasks.

Corresponding author: Luis Carlos dos Santos Coutinho Retondaro, *luis.retondaro@cefet-rj.br*

***Index Terms*—Document layout, Hierarchical clustering, Interaction design.**

I. Introdução

Documentos multimídia são compostos a partir de uma coleção de vários ativos de mídia, tais como imagens, textos e filmes. Esses ativos precisam ser dispostos no espaço (por exemplo, numa página) e podem também ser dispostos no tempo, a fim de elaborar composições, tais como animações animações ou apresentações interativas. Em uma composição, elementos de mídia são dispostos de forma semanticamente coerente, ou seja, sua disposição no espaço e no tempo tende a seguir a linha da narrativa [1]. Angelides [2] sugere que personalizar conteúdo multimídia é um processo trabalhoso, que envolve extrair e modelar informações estruturais sobre o conteúdo, exigindo que os autores especifiquem diferentes tipos de informações em diferentes níveis [3], [4].

Em geral, o paradigma de interface de um sistema de autoria de apresentação é estrutural. Isto significa uma ênfase no arranjo dos elementos e no modo de ativação ou exibição de cada um. Uma preocupação especial da abordagem baseada em estrutura é a composição da sincronização, isto é, uma coleção de ativos de mídia necessita ser especificada de tal forma que estes possam ser apresentados de maneira coordenada. Tipicamente, a sincronização pode ser baseada em: relacionamentos de tempo, ordenação estrutural relativa, um conjunto de restrições (como ter uma mídia A repetida durante a exibição de B) ou com base em composição atemporal (na qual grupos de ativos são identificados como pertencentes um ao outro sem especificar relações de tempo predeterminadas entre os grupos) [3].

Nesse processo de layout, a tarefa essencial é selecionar e agrupar elementos semelhantes. A forma mais tradicional de realizar esta tarefa utiliza alguma interface baseada no paradigma WIMP (Window, Icon, Menu, Pointer) para selecionar e agrupar elementos através de mecanismos de clicar e arrastar. Entretanto, as interfaces tradicionais podem ter um desempenho insatisfatório, pois exigem vários comandos individuais, e as tarefas se tornam tediosas porque são baseadas em ações repetidas para realizar uma única operação.

Outro problema é que, após a identificação dos grupos, o usuário deve explicitamente relacioná-los para destacar a

semântica de toda estrutura do agrupamento. Em geral, usa-se alguma estrutura visual, como listas ou árvores recolhíveis, onde as folhas representam os itens e os nós internos representam os grupos.

Além desses problemas, mídias brutas, como desenhos a mão livre ou notas manuscritas, são tratadas como imagens digitalizadas individuais. Assim, para utilizar partes deste material, primeiro é necessário empregar algum software de segmentação de imagens, seguido da coleta manual das partes úteis, o que diminui sensivelmente a eficiência de todo o processo de autoria.

Neste artigo, propomos vários remédios para esses problemas. Acima de tudo, sugerimos que a proximidade espacial é uma metáfora mais eficaz para transmitir relações semânticas entre elementos de conteúdo do que agregações simbólicas na forma de árvores ou listas recolhíveis. Apresentamos um conjunto de técnicas de interação para construir tais hierarquias, organizá-las em grupos semânticos, bem como organizá-las espacial e temporalmente na forma de *slides*.

Empregamos o termo *patch* para nos referirmos a cada elemento gráfico, ou seja, cada componente conexa advinda da segmentação da imagem original de entrada. Na Figura 1, por exemplo, há uma série de patches correspondentes a marcas gráficas de texto e um outro patch correspondente a uma figura de coelho. Claramente, os patches referentes ao texto têm uma hierarquia de agrupamento natural, que é definido pela distância entre as marcas gráficas.

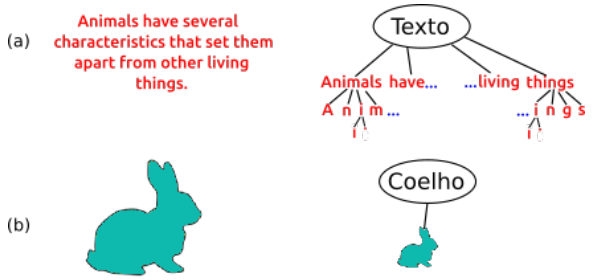

Fig. 1. Esquerda: *patches* formados por cada marca gráfica, a partir da segmentação de imagem - (a) texto e (b) coelho. — Direita: Hierarquia de agrupamento natural correspondente a (a) e (b).

Para indicar graficamente o arranjo hierárquico dos elementos, propomos o emprego de curvas fechadas suaves criadas como isocontornos de uma classe de funções implícitas chamadas *metaballs*. Uma vez que essas funções estão relacionadas à distância de uma forma central, elas fornecem uma representação natural de elementos agrupados de acordo com sua proximidade. A Figura 2 mostra o agrupamento natural em quatro níveis distintos de hierarquia.

Este artigo descreve o design da interface proposta, incluindo como executar tarefas típicas, bem como os algoritmos e estruturas de dados necessários para sua implementação.

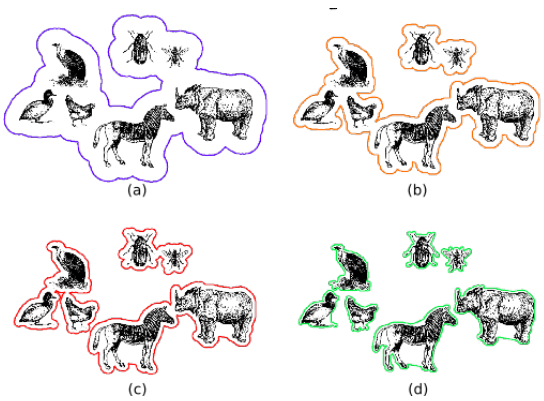

Fig. 2. Exemplo de visualização de agrupamentos semânticos em quatro níveis de hierarquia. A curva azul (a) indica o nível mais abrangente contendo todos os patches com figuras de animais; em (b), as duas curvas laranjas indicam os dois filos (cordados e artrópodes); as três curvas vermelhas (c) indicam as classes (aves, mamíferos e insetos); em (d) as curvas verdes representam o nível mais baixo e contornam cada animal em separado.

II. Trabalhos relacionados e fundamentos

Vários métodos de controle de layout espacial têm sido propostos na literatura. Um trabalho seminal interessante relacionado à organização de conteúdos em níveis de detalhe é o sistema PAD++ [5]. Ele usa um paradigma de interface baseado em *zoom*, onde os conteúdos que são dispostos em um plano conceitualmente infinito, de modo que os elementos semanticamente conectados podem ser colocados juntos, mas longe de outros grupos semânticos. A ideia é particularmente interessante como sugestão de organização hierárquica. Uma técnica de interação chamada *the vacuum* é descrita em [6], onde um *widget* circular especial é proposto para criar manipuladores para representar objetos distantes na tela - a ideia é criar e usar esses manipuladores ao invés de deslocar o mouse para a região da tela onde esses objetos se encontram. Outra ideia relacionada, os *ninja cursors* [7] permitem a interação com objetos distantes através de operações de *zoom in* e *zoom out* controladas implicitamente através da velocidade de deslocamento do mouse. Podemos ainda citar outras abordagens para melhorar a interação com objetos distantes [8], [9], subgrupos de janela [10], interfaces de zoom [5], [11] e marcos artificiais [12].

A. Bubble Clusters

O modelo de interação conhecido como *Bubble Clusters* [13] é uma das principais inspirações da presente proposta. Ele foi originalmente proposto para ajudar os usuários a organizar coleções de ícones na tela para que eles possam ser agrupados e reagrupados espacialmente usando o mouse. O artigo também descreve um aplicativo em que traços à mão livre podem ser manipulados. A ideia é usar a proximidade entre os vários objetos na tela, reconhecendo automaticamente os objetos próximos como grupos. Esses grupos são sinalizados visualmente pelo traçado de uma curva em torno deles semelhante a uma bolha de sabão.

Para compor as bolhas, o método computa uma função radial quadrática $f(r)$ baseada no conceito de *metaballs*. Cada objeto é associado a um ponto que define um campo potencial ao redor dele que diminui com a distância e desaparece além de um determinado limite. A curva em si é simplesmente um isocontorno deste campo que é traçada usando o método *Marching Squares* [14]. Se houver vários objetos no cluster, os campos potenciais de objetos próximos são somados para gerar um limite curvo suave. A função $f(r)$ satisfaz $f(r_0) = 1, f(r_1) = f'(r_1) = 0$, onde r_0 é o raio da bolha de um objeto isolado e r_1 é o tamanho limite (threshold) do campo (vide Fig. 3).

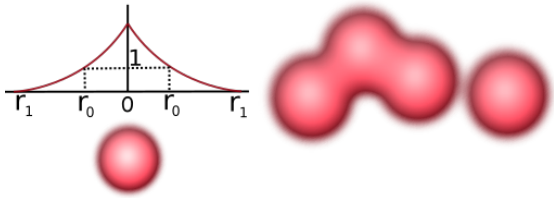

Fig. 3. Campo potencial [13]. (esquerda) Função f(r) que compõe o campo potencial de uma bolha simples; (direita) Exemplo da soma de campos potenciais.

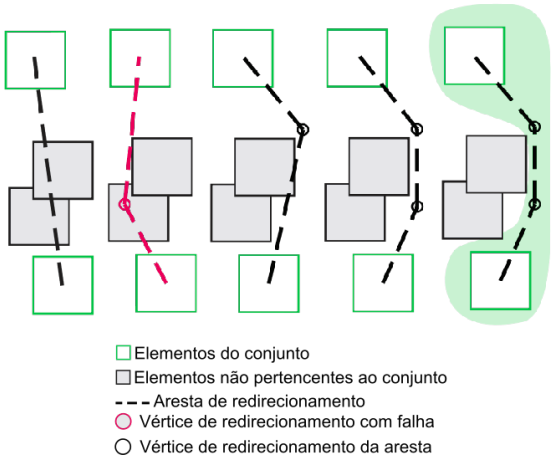

Fig. 4. (Da esquerda para direita): Uma aresta virtual passando por um elemento é detectada. Um novo ponto de controle é criado em um canto da caixa delimitadora do obstáculo e o teste é repetido. Quando o teste falha, o canto diagonalmente oposto é usado e nenhum obstáculo é encontrado. Pontos de controle adicionais são criados nos cantos para direcionar a aresta ao redor do obstáculo. O conjunto final de arestas virtuais contribui para o cálculo da energia, permitindo que o contorno definido evite os obstáculos e permaneça conectado.

Essa abordagem de agrupar ícones como uma bolha não funciona apenas como uma indicação visual para indicar a existência de um grupo, mas também como um identificador para interagir com o grupo. A operação mais básica é arrastar um grupo de ícones dentro de uma distância limite: o usuário pode arrastar um grupo inteiro arrastando a bolha.

B. Bubble Sets

Bubble Sets [15] é uma técnica de visualização de dados para destacar conjuntos de pontos que estão irregularmente distribuídos em um gráfico ou mapa existente. Como os bubble clusters, ela usa uma curva em forma de bolha para circundar os elementos pertencentes ao mesmo grupo, mas não exige que estejam próximos uns dos outros. Os autores propõem uma abordagem heurística para construir bolhas esteticamente agradáveis. A ideia é construir primeiro uma árvore geradora conectando todos os elementos do conjunto de forma que não apenas os pontos (vértices) sejam usados como centros das *metaballs*, mas também algumas arestas selecionadas que são necessárias para manter a curva conectada. A conexão do contorno é garantida por meio de arestas virtuais que contribuem para a distribuição de energia do conjunto. Da mesma forma, algumas heurísticas são empregadas para garantir que as curvas de bolha não se cruzem ou que envolvam elementos não pertencentes ao conjunto como ilustra a Figura 4.

C. Segmentação da imagem

Os sistemas de autoria multimídia empregam diversos tipos de materiais gráficos como entrada e tendem a incluir ferramentas auxiliares para composição, desenho e edição de imagens. Aqui, consideramos essas ferramentas como fornecidas e nos concentramos nas técnicas para organizar sua produção. Uma vez que as imagens digitais - renderizadas ou digitalizadas - podem ser vistas como denominador comum, propomos tratar todos os tipos de conteúdo simplesmente como grupos de *pixels*. Portanto, empregamos alguns meios para extrair marcas gráficas de documentos apresentados como imagens, ou seja, recorremos a técnicas de *segmentação de imagem* para separar os *pixels* do primeiro plano (marcas gráficas) dos *pixels* do fundo. Os *patches* são formados a partir destes segmentos de imagem.

D. Aproximação da forma

Em ambientes interativos, caixas delimitadoras mínimas são frequentemente usadas para agrupar elementos gráficos. Essas caixas podem ser organizadas como *Bounding Volume Hierarchies* (BVH), estruturas de dados hierárquicas usadas em muitas aplicações para modelar relações espaciais, como detecção de colisão [16], [17], correspondência de imagens [18], anonimização de dados [19], métodos não-paramétricos [20] e indexação espacial [21], [22]. Para superar o desafio de encontrar o par mais próximo de pontos entre dois patches, bem como outras consultas espaciais recorremos a uma classe de BVHs conhecidas como *ball trees* [23]. *Ball trees* permitem representar uma forma complexa em vários níveis de detalhe usando círculos delimitadores. Usando níveis rasos da árvore, é possível ter uma aproximação da forma do objeto. Além disso, encontrar o par mais próximo de pontos entre dois objetos representados por ball trees leva tempo $O(\log^2 n)$ em vez de $O(n^2)$. Tipicamente, Ball trees otimizadas e com performance eficiente para as consultas propostas contêm

um número reduzido de bolas de entrada, bem como uma área total minimizada.

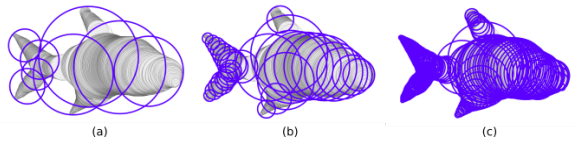

Fig. 5. Exemplo de uma forma representada por uma *ball tree* nos níveis 2, 3 e 4 (a-c)

Weller, et al. [24] propuseram a implementação de uma abordagem baseada em *circle packing* para minimizar de forma eficiente a quantidade de bolas de entrada na construção da sua estrutura denominada *Inner sphere trees*.

Outra ferramenta convenientemente utilizada aqui para representar objetos no espaço digital é o eixo medial - *Discrete Medial Axis* (DMA) - pois além de ajudar a minimizar o número de bolas na composição da ball tree, é reversível, ou seja, a partir de seus pontos, podemos reconstruir a forma original [25].

Na verdade, o DMA é definido como o conjunto de pixeis centrais do círculo máximo que cobrem a forma do objeto. Então, um círculo máximo é um disco contido na forma não inteiramente coberta por outro disco contido na forma [25]–[28] (vide Figura 6).

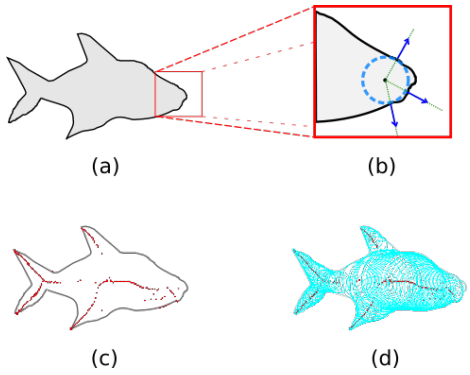

Fig. 6. Aproximação da forma baseada no eixo medial. (a) Forma original. (b) Detalhe da definição do DMA. (c) Pontos do DMA. (d) Representação do objeto pela união das bolas.

Outra inspiração para nossa abordagem foi o trabalho de Broutta, et al. [25] que implementou um método para construir BVHs de esferas - também conhecido como *sphere-trees* - em tempo linear onde os conjuntos de esferas em cada nível são obtidos por uma extração discreta do eixo medial em diferentes níveis de resolução de objeto, usando uma abordagem piramidal regular e controlando de forma eficiente a propagação do erro e sua distribuição em cada nível. A Figura 7 ilustra os principais estágios desse processo.

E. Transformada da Distância

Um método clássico para calcular distâncias a um objeto em uma imagem digital foi proposto por Borgefors [29] e é conhecido como transformada de distância (DT). Em

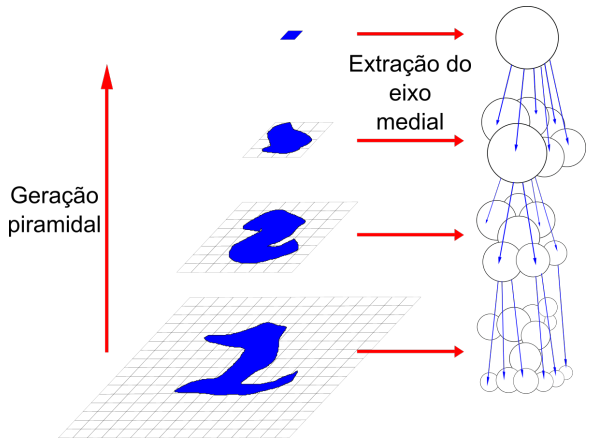

Fig. 7. Principais estágios da construção da *sphere tree* [25], onde os conjuntos de esferas em cada nível são obtidos por uma extração de DMA em diferentes níveis de resolução do objeto, em uma abordagem piramidal regular.

imagens binárias, DT e a extração de esqueleto geométrico com base no eixo medial (DMA) são ferramentas clássicas para a análise de formas [30]–[32]. As transformadas de distância usam uma discretização de grade ou um objeto para resolver o problema de encontrar a distância de um determinado ponto a uma forma e podem ser calculadas de forma eficiente com GPU-shaders usando um algoritmo chamado *jump flooding* (JFA) [33]. Além disso, um diagrama de Voronoi pode ser usado para construir um histograma das distâncias entre os vários segmentos de uma imagem. Desta forma, os intervalos mais frequentes deste histograma podem ser usados para estimar os limites de distância. Outras abordagens mais rápidas do que JFA foram implementadas [34], mas com menor precisão de cálculo. Experimentos empíricos que realizamos, resultaram em um baixo desempenho do JFA em GPU para reconstrução e manipulação da estrutura a taxas iterativas.

Em outro trabalho destacado, Coeurjolly et al. [35] apresentam algoritmos de tempo ótimo para resolver a transformação reversa da distância euclidiana e os problemas de extração reversível do eixo medial para imagens d-dimensionais, além do processo de filtragem do eixo medial que permite controlar a qualidade da forma reconstruída.

III. Interface e Usabilidade

Esta seção descreve os principais recursos da interface e seu comportamento.

A. Estado Inicial

Para o trabalho de composição do layout, assumimos que o estado inicial já dispõe na tela os patches com uma sugestão de agrupamento. Com base na imagem de entrada, as relações de distância originais são mantidas e os patches são posicionados adequadamente. Da mesma forma que Watanabe [13], os resultados do agrupamento são mostrados como bolhas ao redor de cada grupo de patches (Figura 8). Apesar da semelhança, neste trabalho,

construímos um modelo de interação cuja abordagem difere daquela de Watanabe [13] em dois aspectos principais. A primeira é que os elementos a serem agrupados não são pontos, mas objetos com formas irregulares, como fragmentos de imagens ou parágrafos formatados. Na verdade, o artigo original sobre *bubble clusters* também se refere a essa barreira de desempenho ao descrever o aplicativo para agrupar desenhos e notas à mão livre. A segunda é que o agrupamento é realizado em vários níveis de detalhes.

A partir deste estado inicial, a interface oferece algumas operações de composição descritas a seguir.

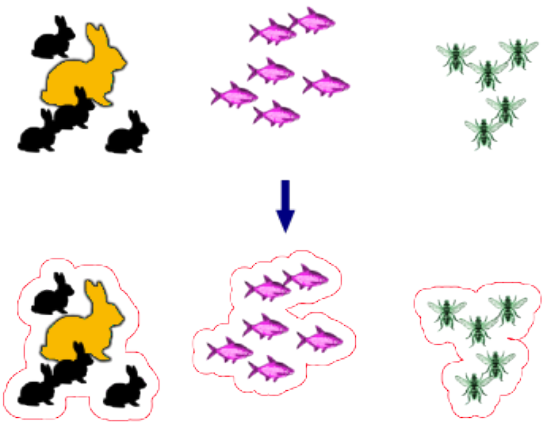

Fig. 8. Layout espacial de objetos. Topo: agregações de objetos representam sua estrutura de grupo. Embaixo: as bolhas são sinalizações dos grupos.

B. Reagrupamento

A hierarquia dos grupos é construída implicitamente, a partir da relação de proximidade entre objetos cujo padrão é definido pelo usuário, que determina um limiar de distância mínima (*threshold*) para cada nível. Para facilitar o reagrupamento automático, nossa interface estabelece 4 faixas de distância, que foram pré-estabelecidas empiricamente: $[0, 1.6)$, $[1.6, 5.1)$, $[5.1, 15.1)$ e $[1.5, \infty)$. Desta forma, o ajuste do nível de distância é feito através de duas operações básicas. A primeira é a escolha da faixa corrente. Dependendo da faixa selecionada tem-se uma aproximação mais direta do detalhamento pretendido. A segunda opção é a modificação da distância mínima para um melhor refinamento do nível de detalhe do agrupamento. Por exemplo, a Figura 9 mostra um contexto de uso em três etapas. Inicialmente, o usuário seleciona a primeira faixa com uma distância mínima igual 0, onde cada patch é também um cluster isolado. À medida que o usuário aumenta a distância mínima, no segundo momento, pode-se observar os objetos ainda no mesmo nível de agrupamento, porém a bolha ao redor dos patches indica uma nova abrangência. Na última etapa do processo, o usuário obtém o próximo nível de hierarquia, onde os dois patches estão agrupados.

A Figura 10 mostra a clusterização automática produzida a partir do parâmetro de distância mínima em cada nível. Como entrada temos uma página de texto com

Fig. 9. Agrupamento implícito de patches. O usuário pode controlar os diferentes níveis da clusterização.

poucas figuras. Para facilitar a visualização, diminuímos a opacidade da imagem digitalizada e exibimos três níveis de clusterização, simultaneamente, em cores diferentes: niveis 1 (curva vermelha), 2 (curva azul) e 3 (curva verde). Note que o nível 1 agrupa os patches em palavras, o nível 2 em parágrafos e o nível 3 em blocos de texto.

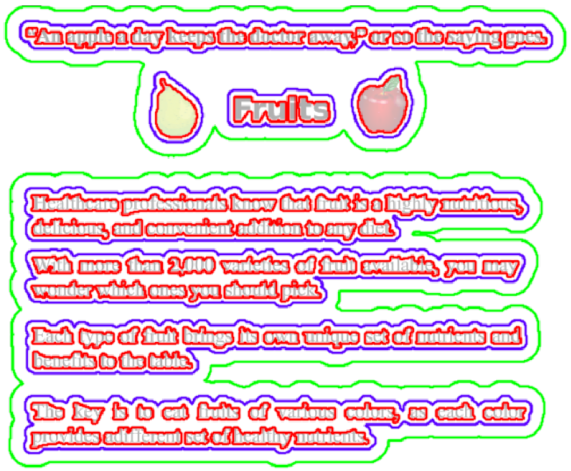

Fig. 10. Exemplo da geração de curvas para 3 níveis de agrupamentos distintos.

C. Posicionamento

Durante a manipulação, o usuário poderá também arrastar um elemento no nível corrente, tratando-o como um membro de um determinado cluster. Neste caso, este elemento pode mover-se de um cluster para outro ou pode ser mesclado com outros clusters nesse nível.

No sistema proposto realizamos automaticamente a detecção de colisão entre os elementos, fazendo com que o ato de arrastar um objeto, empurre o outro que está no seu caminho. Isto evita a sobreposição dos elementos e mantém a clareza quanto às relações de proximidade entre eles. Porém, na maioria das operações de ajuste de layout, o que o usuário precisa é poder mover o objeto em qualquer direção sem a preocupação de deslocar os demais elementos. Por exemplo, a Figura 11 mostra um deslocamento que desajusta o texto "Orange" no nível 0. Para evitar estas situações, a interface proposta prevê duas alternativas: (1) o usuário pode mudar o nível da clusterização, assim os objetos empurrados serão deslocados em grupo, o que parece razoável, já que são segmentos semânticos distintos; (2) o usuário pode desativar a detecção de colisão. Quanto a este segundo aspecto, é importante ressaltar que ao deslocar um objeto sobre outros não haverá colisões. Porém, ao término do movimento, se o objeto sobrepuser outro, eles

se deslocarão automaticamente para evitar a sobreposição final.

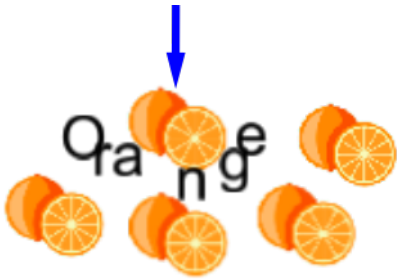

Fig. 11. Deslocamento de patches no nível mais baixo após colisão.

A sugestão visual de um novo cluster através do desenho das curvas em tempo real, preserva o processo cognitivo da interface e mantém a usabilidade. Entretanto, para acelerar o processo, a interface também permite ao usuário desativar o desenho das curvas, se necessário. Além disso, o sistema de clusterização automática é executado em intervalos determinados e só é acionado quando: (a) diretamente por comando do usuário quando ele deseja confirmar a sugestão proposta; (b) quando algum parâmetro da interface é modificado, tal como: distância mínima, faixa de distância, ativar/desativar detecção de colisão e desenho de curvas.

D. Separação

Para algumas situações específicas determinadas pelo usuário, pode ser mais fácil separar um ou mais elementos de um cluster sem ter que modificar a distância limite, e ainda, sem ter que mover os objetos. Nessas situações o usuário aciona o comando *separar*.

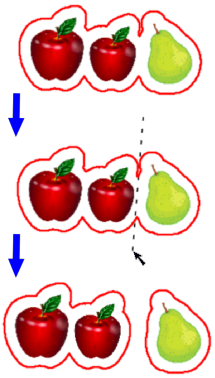

Fig. 12. Separação de um objeto em um cluster.

Na interface proposta, implementamos um mecanismo baseado em traços, onde o usuário pode rabiscar para executar comandos. O comando separar é executado, traçando-se um segmento de reta que atravessa o cluster no ponto de interesse, determinado pelo usuário. Os objetos que estiverem localizados em lados opostos da linha serão separados, ou seja, novos clusters serão criados, considerando o intervalo que será gerado entre estes objetos. A Figura 12 ilustra esse processo.

E. Utilitários

Em adição às funções básicas, a interface do nosso sistema permite que o usuário desfaça até 10 ações equivocadas, através do comando *desfazer*. Além disso, é possível reiniciar todo processo acionando o comando *reset*.

Outro mecanismo clássico que implementamos em nosso sistema foi o comando *copiar e colar*. Apenas com o uso do botão do meio do mouse, o usuário pode clicar para selecionar o objeto enquanto o copia para a área de transferência. A seguir, com o mesmo comando é possível "colar" o objeto na posição corrente do mouse.

IV. Implementação

Os algoritmos utilizados combinaram as ideias e conceitos fundamentais dos trabalhos descritos na seção II com algumas adaptações importantes.

Após digitalizar o documento, realizamos a segmentação na forma descrita na seção II-C. Com os patches formados, realizamos a extração do esqueleto medial.

Adaptamos os algoritmos eficientes implementados por Coeurjolly, et al. [35]: REDT, que associa o cálculo da transformada de distância euclidiana reversa com a extração do eixo medial discreto reduzido (RDMA). Tanto o cálculo do REDT quanto a extração do RDMA obtiveram um custo de computação da ordem de $O(n^2)$ que é baseada na construção do diagrama de Voronoi com pesos [36]. Implementamos REDT e RDMA a fim de obter um conjunto final menor de bolas, uma vez que a representação reversível da forma é mais compacta e gera menos pontos.

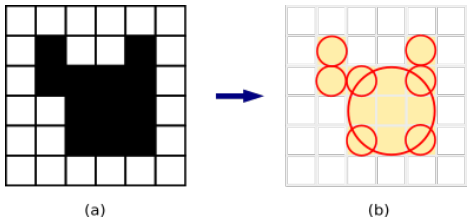

Fig. 13. Ilustração do algoritmo REDT/RDMA. (a) forma orginal — (b) conjunto mínimo de bolas necessárias para reconstruir a forma.

Seja $B = \{B_i\}$ o conjunto de bolas (círculos) obtidos a partir da extração do eixo medial e r_i seus raios. B pode não conter a cobertura exata da região (Figura 14). Nessa etapa, utilizamos uma abordagem conservadora para garantir a condição de cobertura da forma do objeto. Adicionamos um valor mínimo à extensão do raio e cada raio é estendido por (r'_i), suficiente para cobrir a região da caixa delimitadora de cada bola, de modo que, $r'_i = r_i\sqrt{2}$.

Consideramos que os círculos formados por este procedimento serão as bolas de entrada para a construção da ball tree de cada patch que, por sua vez, representam a forma do objeto.

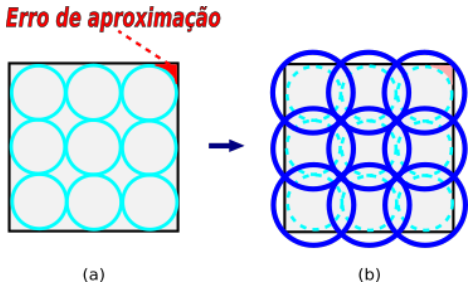

Fig. 14. Evitando erro de aproximação de cobertura. (a) Detalhe da região. (b) Simples extensão do raio.

Utilizamos o algoritmo Q [37] com $\alpha = 0,5$ para a construção das ball trees, pois garante a eficiência das consultas de proximidade utilizadas tanto na composição dos clusters, que são formados automaticamente, quanto nas operações de detecção de colisão e separação de elementos.

Para implementar as curvas de nível usamos um algoritmo simples que consiste em gerar uma máscara a partir das folhas das *ball trees* cobrindo a forma do objeto, onde o raio de cada círculo é aumentado proporcionalmente ao limite de distância requerido. Assim, obtemos uma curva de contorno que guarda um elevado grau de precisão de distanciamento da forma original. Estas curvas servirão como indicação visual dos clusters.

Além de construir as curvas, nosso sistema reconhece estruturas implícitas no ambiente 2D e analisa suas informações espaciais e relações de proximidade para então propor um layout automático. Este é o algoritmo de clusterização, cujo desempenho é inversamente proporcional à quantidade de *patches* na cena de trabalho. As comparações são realizadas para cada |patch e a complexidade do algoritmo é quadrática. Entretanto, a usabilidade foi preservada, porque aceleramos o processo interativo considerando dois aspectos principais: (1) além dos *clusters* que tiveram sua posição original alterada, somente aqueles que fazem interseção com estes são pesquisados; (2) a interface visual oferece uma sugestão de agrupamento ao usuário, porém a confirmação dos agrupamentos só é efetuada em intervalos determinados, como mencionado anteriormente.

V. Experimentos

De forma a avaliar a usabilidade e desempenho da interface descrita, conduzimos um conjunto de experimentos diferentes, onde propusemos tarefas de layout para um grupo de usuários de teste. Assim, medimos o tempo gasto para a execução de cada tarefa, bem como o número de comandos executados por eles. Todas as tarefas tinham um objetivo claro e direto: modificar o layout de um documento de entrada, reajustando a posição dos elementos para um novo layout pré-determinado.

O protótipo da interface foi construído em Javascript e testado em um navegador Chrome v92.0.4, sendo disponibilizado para um grupo de 16 usuários de teste que realizaram as tarefas em suas próprias estações de trabalho. A Figura 15 mostra a tela da aplicação durante um exercício para a execução de uma dessas tarefas. O perfil selecionado

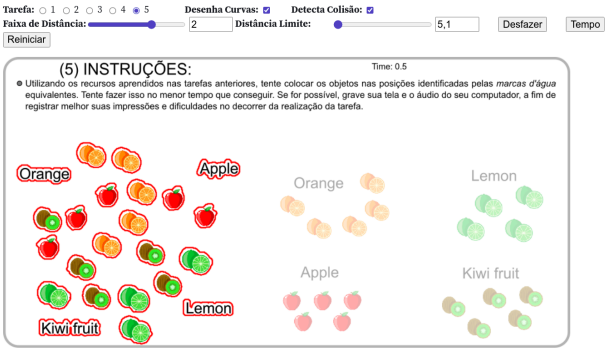

Fig. 15. Exemplo da interface durante o exercício de preparação para a execução de uma das tarefas.

foram de usuários frequentes ou especialistas em softwares de edição gráfica. A interface que serviu de veículo de comparação foi o software de desenho vetorial conhecido do grupo selecionado de usuários: o Inkscape v1.11 [38]. A interface de seleção e agrupamento de elementos desse software segue o método tradicional usado em aplicações desse gênero (veja Seção I).

Cada usuário teve suas tarefas cronometradas e gravadas em video. Uma única tela continha espaço para os dois layouts: o original e a sugestão proposta foram alocados lado a lado, para evitar mudança ou rolamento de tela, preservando o foco do usuário.

Os usuários de teste participaram de um breve treinamento de uso da nova interface, sem limite de tempo, para testar e aprender os comandos: selecionar e arrastar objetos, alternar entre níveis de agrupamento, ativar a clusterização, modificar a distância mínima no nível corrente, copiar e colar clusters, separar objetos de um cluster e detectar colisão. Nenhuma dificuldade foi relatada nas 4 tarefas de treinamento, exceto 3 comentários sobre o comando "copiar e colar". Segundo esses relatos, a dificuldade encontrada nesta ação foi a identificação do ponto exato do alvo.

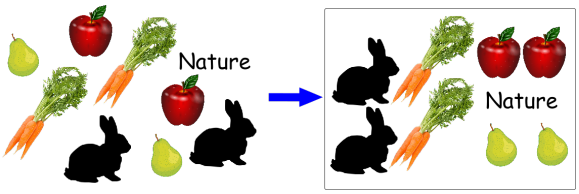

Fig. 16. Tarefa T1 - Imagem de entrada e layout sugerido.

Nesse contexto, 3 tarefas específicas de layout foram preparadas para serem executadas no menor tempo possível. Essas tarefas, denominadas respectivamente por T1, T2 e T3, foram estabelecidas com grau de dificuldade distintos. Em T1, o documento de entrada contém somente imagens de alguns ícones (Fig. 16), contendo um total de 14 patches. Em T2, há mais imagens do que texto, num total de 47 patches (Fig. 17). Note que em T3 (Fig. 18), com 259 patches, há maior quantidade de texto, porém o alvo exige pouca modificação. Todas as imagens de entrada

foram digitalizadas com a resolução máxima de 520×400 a 300 DPI.

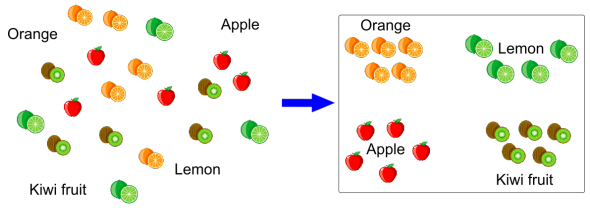

Fig. 17. Tarefa T2 - mais imagem do que texto.

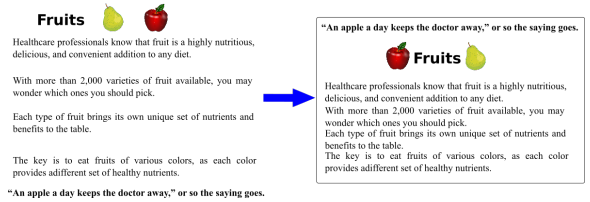

Fig. 18. Tarefa T3 - mais texto, porém menos modificações sugeridas no layout.

Antes da apuração definitiva dos resultados, realizamos o acompanhamento em tempo real da execução das tarefas com 3 usuários experientes. As observações destes usuários especiais, ao longo da execução da tarefa, forneceram parâmetros importantes para algumas melhorias na interface, como posicionamento de alguns ícones, filtro para detecção de colisão e mecanismo usado para separação de objetos.

Inicialmente, analisamos o tempo de execução das tarefas. A Tabela I compara nossa abordagem de clusterização hierárquica com o método tradicional. Note que o tempo médio de execução utilizando o método tradicional foi 15%, 22% e 13% mais rápido em T1, T2 e T3, respectivamente. Dado que todos são especialistas em Inkscape, é razoável esperar que o tempo em uma interface nova seja maior, mesmo que o treinamento tenha sido eficiente.

TABELA I
TEMPO MÉDIO DE EXECUÇÃO DAS TAREFAS (EM SEGUNDOS)

Interface	T1	T2	T3
Clusterização Hierárquica	28	112	39
Tradicional	24	88	34

Apesar do tempo de execução mais elevado, quando analisamos os comandos dos usuários, observamos que o fluxo de trabalho da interface proposta é eficiente e fornece uma perspectiva satisfatória. São dois aspectos importantes que nos conduzem à essa conclusão. Primeiro, o volume de ações e comandos emitidos pelos usuários. Esse total foi medido pela quantidade de cliques com o mouse. A Tabela II mostra a quantidade média de comandos executados para cada tarefa.

Observe que os usuários executam menos comandos com a interface proposta. Por exemplo, na tarefa T3, a clusterização hierárquica facilita bastante a seleção de parágrafos inteiros de maneira mais intuitiva e imediata, sem a necessidade de criar uma caixa envolvente. Esta

TABELA II
QUANTIDADE MÉDIA DE COMANDOS EXECUTADOS POR TAREFAS

Interface	T1	T2	T3
Clusterização Hierárquica	10	27	9
Tradicional	12	30	15

observação foi relevante, pois onde há mais texto a dificuldade de seleção aumenta, quer seja pelo tamanho reduzido dos objetos (letras), quer pela sua disposição longitudinal.

O segundo aspecto importante a ser considerado quanto à eficiência da interface são os tipos de comandos executados por cada usuário. Tanto na interface proposta quanto na tradicional, há maneiras diferentes de executar comandos e se obter o mesmo resultado. Alguns usuários, propositadamente ou não, escolheram o caminho mais complexo ou menos trivial. Especificamente na tarefa T2, usando o método tradicional, o trivial seria selecionar os elementos um a um. Ao invés disso, 4 usuários decidiram: (a) selecionar os objetos do mesmo grupo, (b) movê-los na direção desejada e (c) selecionar individualmente cada um, ajustando-o para a posição alvo. Este processo parece mentalmente mais organizado, pois considera o significado dos objetos e a relação destes com sua disposição no layout. Entretanto, toda a rotina percorre um caminho mais demorado e tedioso. Por outro lado, se considerarmos o sentido inverso, ou seja, documentos que já estejam montados considerando a relação de proximidade entre objetos de um mesmo grupo semântico, a interface proposta seria mais eficaz, porque já considera conceitualmente estas relações. Assim, tanto do ponto de vista da execução, quanto do processo cognitivo do usuário, nos parece mais simples agrupar para depois ajustar ao *layout*.

As execuções das tarefas T1 e T3 foram mais uniformes entre os usuários. Porém, uma outra divergência foi observada na tarefa T2: houve um usuário que utilizou o comando de corte para separar todos os elementos primeiro, para só depois movê-los individualmente. Esta iniciativa parece indicar que o usuário não percebeu que poderia apenas alternar para um nível mais baixo na hierarquia.

Curiosamente, nenhum usuário utilizou o comando "copiar e colar", talvez pela dificuldade mencionada anteriormente ou apenas por considerar o comando inadequado.

Outra análise importante está relacionada ao desempenho do motor da interface que contém 3 algoritmos principais: (A1) desenho de todos os patches, (A2) confecção e desenho das curvas de agrupamento e (A3) reclusterização de todo conjunto de patches.

TABELA III
TEMPO MÉDIO DE EXECUÇÃO DE CADA ALGORITMO (EM MILISSEGUNDOS)

Tarefas	A1	A2	A3
T1	0.1	67	3.5
T2	0.3	58	22
T3	1.0	110	270

Claramente, o tempo médio de execução de cada algoritmo, durante o processo interativo, depende da imagem

de entrada. A Tabela III compara essas etapas para a interface proposta. O algoritmo $A3$, por ser quadrático, é o que mais se ressente quando número de *patches* é elevado. O maior gargalo está no algoritmo $A2$, que precisa ser executado a cada quadro durante o arraste do mouse. Ainda que este seja um tempo aparentemente irrisório, pode diminuir a taxa de exibição a um valor inferior a 10 FPS.

VI. Conclusões

Este trabalho apresenta uma nova interface para sistemas de autoria de conteúdo multimídia. Uma nova abordagem propõe a manipulação direta para ajudar a estruturar hierarquias semânticas de elementos gráficos. Os clusters de objetos e seus níveis de organização são gerados de forma implícita e sinalizados visualmente, liberando o usuário da tarefa explícita de agrupamento.

As técnicas empregadas na solução de problemas inerentes ao cenário da aplicação mostraram-se eficazes no apoio à composição de layouts, tanto na preparação quanto no contexto de uso.

Os resultados obtidos demonstram que as funcionalidades implementadas podem ser adaptadas para uso eficiente em softwares de edição de slides e vídeos.

Como trabalho futuro, pretendemos melhorar o algoritmo de geração das curvas de agrupamento, ou ainda, utilizar outro mecanismo visual que possa renderizar mais rápido toda a cena. Também pretendemos incorporar funcionalidades como rotação e alinhamento de objetos.

Referências

[1] A. Scherp, "Authoring of multimedia content: A survey of 20 years of research," em *Semantic Multimedia Analysis and Processing*, CRC Press, 2017, pp. 375–398.

[2] M. C. Angelides, "Multimedia content modeling and personalization," *Encyclopedia of Multimedia*, pp. 510–515, 2008.

[3] D. C. Bulterman e L. Hardman, "Structured multimedia authoring," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, v. 1, n. 1, pp. 89–109, 2005.

[4] M. Jourdan, C. Roisin e L. Tardif, "Constraint techniques for authoring multimedia documents," *Constraints*, v. 6, n. 1, pp. 115–132, 2001.

[5] B. Bederson e J. Meyer, "Implementing a zooming user interface: experience building Pad++," *Softw., Pract. Exper.*, v. 28, n. 10, pp. 1101–1135, 1998.

[6] A. Bezerianos e R. Balakrishnan, "The vacuum: facilitating the manipulation of distant objects," em *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 2005, pp. 361–370.

[7] M. Kobayashi e T. Igarashi, "Ninja Cursors: Using Multiple Cursors to Assist Target Acquisition on Large Screens," em *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, sér. CHI '08, Florence, Italy: Association for Computing Machinery, 2008, pp. 949–958, ISBN: 9781605580111. DOI: 10 . 1145 / 1357054 . 1357201. endereço: https://doi.org/10.1145/1357054.1357201.

[8] M. Whitlock, E. Harnner, J. R. Brubaker, S. Kane e D. A. Szafir, "Interacting with Distant Objects in Augmented Reality," em *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, IEEE, 2018, pp. 41–48.

[9] J. Newn, E. Velloso, M. Carter e F. Vetere, "Multimodal Segmentation on a Large Interactive Tabletop: Extending Interaction on Horizontal Surfaces with Gaze," em *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*, ACM, 2016, pp. 251–260.

[10] D. S. Tan, B. Meyers e M. Czerwinski, "WinCuts: manipulating arbitrary window regions for more effective use of screen space," em *CHI'04 extended abstracts on Human factors in computing systems*, ACM, 2004, pp. 1525–1528.

[11] K. Perlin e D. Fox, "An alternative approach to the computer interface," *Proceedings of Computer Graphics and Interactive Techniques (SIGGRAPH'93)*, pp. 57–64, 1993.

[12] M. S. Uddin, C. Gutwin e A. Cockburn, "The effects of artificial landmarks on learning and performance in spatial-memory interfaces," em *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ACM, 2017, pp. 3843–3855.

[13] N. Watanabe, M. Washida e T. Igarashi, "Bubble clusters: an interface for manipulating spatial aggregation of graphical objects," em *Proceedings of the 20th annual ACM symposium on User interface software and technology*, ACM, 2007, pp. 173–182.

[14] W. E. Lorensen e H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," em *ACM siggraph computer graphics*, ACM, vol. 21, 1987, pp. 163–169.

[15] C. Collins, G. Penn e S. Carpendale, "Bubble sets: Revealing set relations with isocontours over existing visualizations," *IEEE Transactions on Visualization and Computer Graphics*, v. 15, n. 6, pp. 1009–1016, 2009.

[16] M. Kong e Y. Bai, "An Efficient Collision Detection Algorithm for the Dual-Robot Coordination System," em *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, IEEE, 2018, pp. 1533–1537.

[17] H. A. Sulaiman e A. Bade, "Bounding volume hierarchies for collision detection," *Computer Graphics*, pp. 39–54, 2012.

[18] W. Wan e H. J. Lee, "Deep feature representation and ball-tree for face sketch recognition," *Internati-*

onal Journal of System Assurance Engineering and Management, pp. 1–6, 2019.

[19] C. Cheng, L. Xiaoli, W. Linfeng, L. Longxin e W. Xiaofeng, "Algorithm for k-anonymity based on balltree and projection area density partition," em *2019 14th International Conference on Computer Science & Education (ICCSE)*, IEEE, 2019, pp. 972–975.

[20] T. Liu, A. W. Moore, A. Gray e C. Cardie, "New Algorithms for Efficient High-Dimensional Nonparametric Classification.," *Journal of Machine Learning Research*, v. 7, n. 6, 2006.

[21] M. Dolatshah, A. Hadian e B. Minaei-Bidgoli, "Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces," *arXiv preprint arXiv:1511.00628*, 2015.

[22] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," em *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, sér. SIGMOD '84, Boston, Massachusetts: Association for Computing Machinery, 1984, pp. 47–57, ISBN: 0897911288. DOI: 10.1145/602259.602266. endereço: https://doi.org/10.1145/602259.602266.

[23] S. M. Omohundro, *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[24] R. Weller e G. Zachmann, "Inner sphere trees for proximity and penetration queries.," em *Robotics: science and systems*, vol. 2, 2009.

[25] A. Broutta, D. Coeurjolly e I. Sivignon, "Hierarchical discrete medial axis for sphere-tree construction," em *International Workshop on Combinatorial Image Analysis*, Springer, 2009, pp. 56–67.

[26] D. Attali, J.-D. Boissonnat e H. Edelsbrunner, "Stability and computation of medial axes-a state-of-the-art report," em *Mathematical foundations of scientific visualization, computer graphics, and massive data exploration*, Springer, 2009, pp. 109–125.

[27] N. Amenta, S. Choi e R. K. Kolluri, "The power crust, unions of balls, and the medial axis transform," *Computational Geometry*, v. 19, n. 2-3, pp. 127–153, 2001.

[28] T. K. Dey e W. Zhao, "Approximate medial axis as a voronoi subcomplex," em *Proceedings of the seventh ACM symposium on Solid modeling and applications*, ACM, 2002, pp. 356–366.

[29] G. Borgefors, "Distance transformations in digital images," *Computer vision, graphics, and image processing*, v. 34, n. 3, pp. 344–371, 1986.

[30] R. Fabbri, L. D. F. Costa, J. C. Torelli e O. M. Bruno, "2D Euclidean distance transform algorithms: A comparative survey," *ACM Computing Surveys (CSUR)*, v. 40, n. 1, p. 2, 2008.

[31] T. Saito e J.-I. Toriwaki, "New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications," *Pattern recognition*, v. 27, n. 11, pp. 1551–1565, 1994.

[32] F. Chazal e R. Soufflet, "Stability and finiteness properties of medial axis and skeleton," *Journal of Dynamical and Control Systems*, v. 10, n. 2, pp. 149–170, 2004.

[33] G. Rong e T.-S. Tan, "Variants of jump flooding algorithm for computing discrete Voronoi diagrams," em *Voronoi Diagrams in Science and Engineering, 2007. ISVD'07. 4th International Symposium on*, IEEE, 2007, pp. 176–181.

[34] N. Cuntz e A. Kolb, "Fast Hierarchical 3D Distance Transforms on the GPU.," em *Eurographics (Short Papers)*, 2007, pp. 93–96.

[35] D. Coeurjolly e A. Montanvert, "Optimal separable algorithms to compute the reverse euclidean distance transformation and discrete medial axis in arbitrary dimension," *IEEE transactions on pattern analysis and machine intelligence*, v. 29, n. 3, pp. 437–448, 2007.

[36] F. Aurenhammer, "Power diagrams: properties, algorithms and applications," *SIAM Journal on Computing*, v. 16, n. 1, pp. 78–96, 1987.

[37] L. Retondaro e C. Esperança, "Optimized 2D Ball Trees," en, em *Proceedings of the 34th Conference on Graphics, Patterns and Images (SIBGRAPI)*, Los Alamitos: IEEE Computer Society, 2021.

[38] Inkscape. (2021). "Inkscape 1.1.1 released," endereço: http : / / www . inkscape . org (acesso em 01/11/2021).

Luis Carlos dos Santos Coutinho Retondaro é professor do Centro Federal de Educação Tecnológica Celso Sukow da Fonseca - CEFET/RJ, UnED Petrópolis, candidato ao doutorado pelo Programa de Engenharia de Sistemas e Computação da Universidade Federal do Rio de Janeiro - PESC-UFRJ. Por este mesmo programa ele obteve o mestrado na linha de Computação Gráfica no ano de 2008. Em 1990, pela Universidade Católica de Petrópolis - UCP, concluiu o Bacharelado em Ciência da Computação.

Claudio Esperança é engenheiro eletrônico formado pela Universidade Federal do Rio de Janeiro (1980) com mestrado em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro (1990) e doutorado em Ciência da Computação pela Universidade de Maryland (1995). Atualmente é professor titular da Universidade Federal do Rio de Janeiro. Tem experiência na área de Ciência da Computação, com ênfase em Processamento Gráfico, atuando principalmente nos seguintes temas: visualização, modelagem geométrica, animação física, geometria computacional, sistemas de informações geográficas, bancos de dados espaciais.