Linearização de Dados Espaço-Temporais Implementação e Avaliação de Curvas de Preenchimento de Espaço Guiadas por Dados

Gabriel de A. S. Meireles, Nivan R. F. Junior

Centro de Informática
Universidade Federal de Pernambuco (UFPE) – Recife, PE – Brazil

{gasm, nivan}@cin.ufpe.br

Abstract.

Resumo.

1. Introdução

2. Trabalhos Relacionados

3. Metodologia

3.1. Visão Geral

A abordagem proposta fundamenta-se na linearização de estruturas multidimensionais (2D ou 3D) em sequências unidimensionais (1D). Expandindo o conceito de curvas baseadas em contexto (*Context-based Space-Filling Curves*) [1], utiliza-se a técnica de curvas de preenchimento de espaço guiadas por dados (*Data-Driven Space-Filling Curves*) [3]. O núcleo do processamento estabelece um ciclo hamiltoniano que percorre integralmente a grade de pixels, com o objetivo de preservar simultaneamente a coerência espacial, referente a localidade geométrica do caminho, e a coerência dos dados, associada à localidade dos dados no caminho.

Para viabilizar a implementação e a análise experimental, desenvolveu-se uma arquitetura de software híbrida entre C++ e Python. A metodologia trata imagens estáticas no formato $H \times W$ (2D) ou $H \times W \times C$ (3D). Analogamente, vídeos são processados como uma sequência de imagens estáticas do tipo $F \times H \times W$ ou $F \times H \times W \times C$, onde os parâmetros são definidos como:

- F: quantidade de quadros do vídeo;
- H: altura da imagem (número de linhas);
- W: largura da imagem (número de colunas);
- C: número de canais de cor.

O processamento de um vídeo ocorre, inicialmente, pela linearização independente de cada quadro. No entanto, curvas geradas de forma isolada para quadros consecutivos podem apresentar variações topológicas abruptas, prejudicando a coerência temporal. Para mitigar esse problema, o sistema implementa técnicas de alinhamento temporal, visando maximizar a correlação estrutural entre as curvas de frames adjacentes.

Dessa forma, o trabalho atua em duas frentes principais:

- **Imagens:** Implementação modular e reprodutível do método *data-driven* descrito por Zhou et al. [3] para grades regulares, visando a validação dos resultados teóricos;
- **Vídeos:** Investigação e avaliação de técnicas de alinhamento entre quadros para garantir a coerência temporal após a linearização individual de cada quadro do vídeo.

3.2. O método de linearização *data-driven*

A compreensão do método *data-driven* requer a análise preliminar da abordagem (*Context-based*) proposta por Dafner et al. [1]. O conceito central fundamenta-se no mapeamento da imagem original para o seu grafo dual, realizado através da construção de pequenos circuitos, conforme ilustrado na Figura 1.

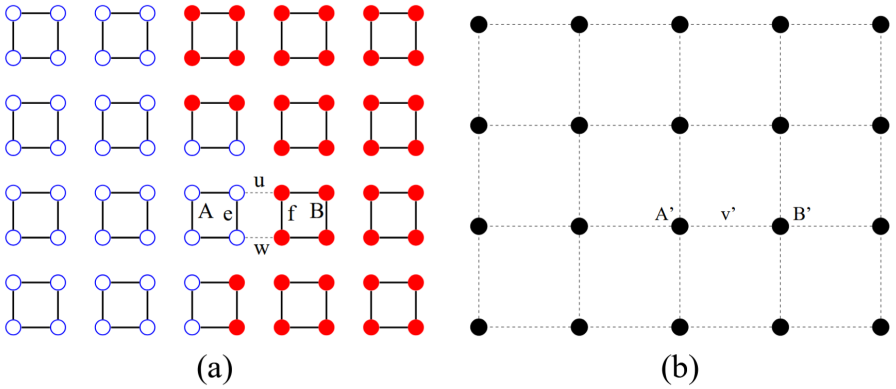

Figura 1. Pequenos circuitos de Dafner et al. [1]

Após esse mapeamento, observa-se que uma árvore geradora no grafo dual equivale a um ciclo hamiltoniano no grafo original. Dessa forma, busca-se definir uma função de peso para as arestas do grafo dual que considere os valores dos pixels da imagem. A partir dessa ponderação, constrói-se uma Árvore Geradora Mínima (MST) no grafo dual para, subsequentemente, recuperar o ciclo hamiltoniano no grafo original. Nesta implementação, adotou-se o algoritmo de Prim para a construção da MST.

A função de custo originalmente proposta é descrita na Equação 1. Nela, $W(C_i, C_j)$ representa o custo de conectar o vértice C_j à árvore geradora, partindo de um vértice C_i já pertencente a ela. As variáveis da equação (u, w, x, y, z, e, f) correspondem aos pesos das arestas no grafo original (ver Figura 2). O peso de cada aresta é definido pela diferença absoluta entre os valores dos pixels em seus extremos. Segundo Dafner et al. [1], para uma aresta $e = (a, b)$, tem-se $w(e) = |a - b|$. Para imagens coloridas (RGB), a diferença é a soma das diferenças absolutas de cada canal: $w(e) = \sum_{k \in \{R,G,B\}} |a_k - b_k|$.

Dessa maneira, a abordagem *context-based* gera linearizações que priorizam a coerência dos dados. Contudo, este método não considera explicitamente a coerência espacial durante a construção da curva. A abordagem *data-driven*, detalhada por Zhou et al. [3], soluciona essa limitação introduzindo uma nova função de custo composta, preservando, no entanto, a estrutura fundamental de mapeamento dual e MST proposta no artigo original [1].

A abordagem *data-driven*, conforme proposto em [3] utiliza a função de custo ponderada descrita na Equação 2. Os componentes da equação são definidos como:

- $N(C_i, C_j)$**:** Custo baseado na coerência dos dados, idêntico à função de Dafner et al. [1] (Equação 1);
- $R_\beta(C_j)$**:** Custo baseado na coerência espacial, definido pela distância euclidiana entre a célula C_j e o seu centróide S_{C_j}, i.e., $||C_j - S_{C_j}||$;
- $\alpha \in [0, 1]$**:** Fator de ponderação. Valores menores de α priorizam a similaridade dos dados, enquanto valores maiores priorizam a regularidade espacial do caminho;
- β**:** Inteiro que define a dimensão $\beta \times \beta$ dos blocos no grafo dual, determinando a posição dos centroides S_{C_j} no centro de cada bloco. Experimentalmente, valores elevados de β tendem a confinar o ciclo hamiltoniano dentro de cada bloco antes de permitir a transição para regiões vizinhas.

Na implementação desenvolvida neste trabalho, α e β foram definidos como hiperparâmetros configuráveis, permitindo o ajuste fino do comportamento da curva e a análise do impacto que a variação desses parâmetros exerce sobre a topologia final. A relação geométrica e estrutural entre os custos de dados $N(C_i, C_j)$ e espacial $R_\beta(C_j)$ é ilustrada na Figura 3.

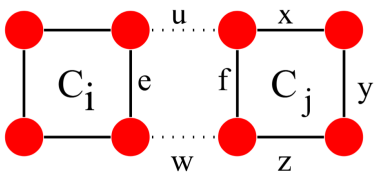

Figura 2. Pesos da função proposta de Dafner et al. [1]

$$W(C_i, C_j) = |u| + |w| + |x| + |y| + |z| - |e| - |f| \tag{1}$$

$$W(C_i, C_j) = (1 - \alpha)N(C_i, C_j) + \alpha R_\beta(C_j) \tag{2}$$

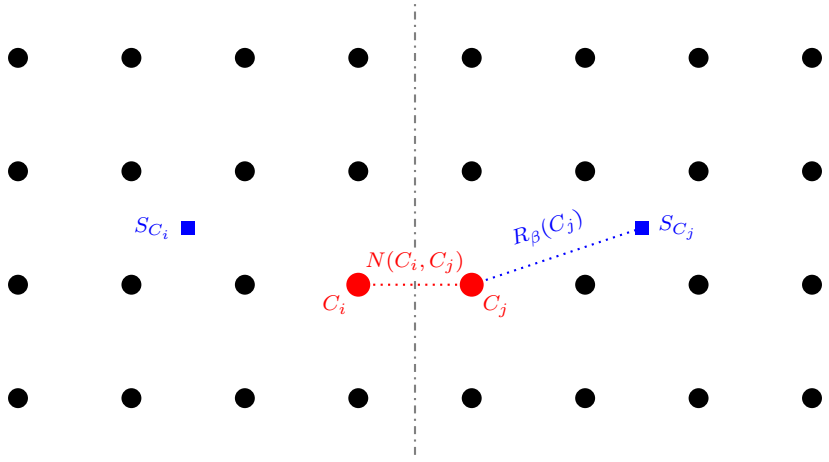

Figura 3. Representação das relações no grafo dual: o custo de dados $N(C_i, C_j)$ e o custo espacial $R_\beta(C_j)$.

3.3. Técnicas de Alinhamento Temporal

Entre dois quadros consecutivos, por mais semelhantes que sejam, pequenas variações nos valores dos pixels podem resultar em linearizações bem distintas. Esse fenômeno prejudica a análise da evolução temporal dos dados, pois quebra visualmente a continuidade entre os quadros. Para mitigar essa instabilidade e promover a coerência temporal na visualização, este trabalho explora duas técnicas de alinhamento. O método fundamenta-se na propriedade de que qualquer rotação cíclica ou inversão de sentido de um ciclo hamiltoniano constitui também uma representação válida de uma curva que cobre integralmente a imagem.

Dessa maneira, o problema de alinhamento consiste em encontrar, para o quadro atual, a rotação e a direção que minimizem uma função de custo objetiva em relação ao quadro anterior. Esse problema pode ser expresso como uma busca pelos parâmetros ótimos definidos na Equação 3. A função f presente nela é definida na Equação 4. Todos os parâmetros e seus significados são explicados a seguir:

- N: Tamanho do vetor da imagem linearizada;
- s: Índice do *cyclic shift* aplicado ao vetor;
- σ: Número binário que indica se o vetor foi invertido ou não;
- V_{t-1} e V_t: Vetores linearizados do quadro anterior e do atual;
- f: Função de custo global que quantifica a dissimilaridade total entre duas sequências;
- g: Função de custo local aplicada elemento a elemento.

$$(s^*, \sigma^*) = \underset{\substack{s \in [0, N-1] \\ \sigma \in \{0,1\}}}{\arg\min} f_\sigma(s) \tag{3}$$

$$f_\sigma(s) = \sum_{i=0}^{N-1} g\Big(V_{t-1}[i],\ V_{\sigma,t}[(i+s) \bmod N]\Big) \tag{4}$$

Neste trabalho, foram avaliadas duas métricas distintas para a função g: a *L1-norm* e a *L2-norm*. Para ambas, são funções facilmente extensíveis para imagens com múltiplos canais.

3.3.1. *L1-norm*

Nesta abordagem, a métrica local g é a diferença absoluta, conforme a Equação 5. Essa escolha mantém a consistência com a função de custo utilizada na construção da MST durante o algoritmo de Prim. Para imagens com múltiplos canais, a função é generalizada pelo somatório das diferenças em cada canal c, correspondendo à distância de Manhattan como destacado na Equação 6.

$$g(a, b) = |a - b| \tag{5}$$

$$g(a, b) = \sum_{c} |a_c - b_c| \tag{6}$$

Devido ao uso do módulo, a busca pela configuração ótima nesta estratégia ocorre de maneira exaustiva. Visto que para cada possível deslocamento é necessário computar a diferença em todos os pixels e canais, a complexidade resultante é $O(C \cdot N^2)$, onde C é o número de canais.

3.3.2. *L2-norm*

Nesta abordagem, a métrica local g é o quadrado da diferença, conforme a Equação 7. Similarmente, essa função pode ser generalizada para imagens com múltiplos canais como apresentado na Equação 8.

$$g(a, b) = (a - b)^2 \tag{7}$$

$$g(a, b) = \sum_{c} (a_c - b_c)^2 \tag{8}$$

Essa formulação permite expandir o termo quadrático da função global que se deseja minimizar e alcançar propriedades algébricas interessantes, como demonstrado na Equação 9:

$$\begin{aligned} f_\sigma(s) &= \sum_{i=0}^{N-1} \Big(V_{t-1}[i] - V_{\sigma,t}[(i+s) \bmod N] \Big)^2 \\ &= \sum_{i=0}^{N-1} V_{t-1}[i]^2 + \sum_{i=0}^{N-1} V_{\sigma,t}[(i+s) \bmod N]^2 - 2 \sum_{i=0}^{N-1} V_{t-1}[i] V_{\sigma,t}[(i+s) \bmod N] \\ &= \sum_{i=0}^{N-1} V_{t-1}[i]^2 + \sum_{i=0}^{N-1} V_t[i]^2 - 2 \underbrace{\sum_{i=0}^{N-1} V_{t-1}[i] V_{\sigma,t}[(i+s) \bmod N]}_{\text{correlação cruzada circular}} \end{aligned} \tag{9}$$

Dessa maneira, observa-se que, para alcançar o mínimo de f_σ, é preciso encontrar os valores de s e σ que maximizem a correlação cruzada circular [2], uma vez que os termos de soma quadrática são constantes e independem da rotação e da direção. Essa propriedade é fundamental pois, dado um $\sigma \in \{0, 1\}$, a correlação pode ser calculada para todo valor de s simultaneamente utilizando a Transformada Rápida de Fourier (FFT) em $O(N \log N)$. Como existem apenas duas orientações possíveis para σ, a complexidade assintótica se mantém. Assim, a complexidade final alcançada é $O(C \cdot N \log N)$, tal que C é o número de canais.

3.4. Detalhes de Implementação

Visando alto desempenho computacional, o núcleo de processamento numérico foi implementado inteiramente em C++20. Para garantir interoperabilidade e facilitar a análise exploratória, as funcionalidades foram expostas ao ambiente Python utilizando a biblioteca `Pybind11`, que atua como uma camada de integração leve.

O ambiente de validação experimental foi construído sobre *Jupyter Notebooks*, permitindo prototipagem rápida e visualização imediata dos resultados. Essa estrutura permitiu avaliar como a variação dos hiperparâmetros impacta o tempo de execução, a correlação entre quadros e a qualidade visual das curvas geradas.

O código-fonte desenvolvido encontra-se disponível publicamente no repositório: `https://github.com/gabmei/TCC`.

4. Experimentos

5. Conclusão

Referências

[1] Revital Dafner, Daniel Cohen-Or e Yossi Matias. "Context-based space filling curves". Em: *Computer Graphics Forum* 19.3 (2000), pp. 209–218. DOI: 10.1111/1467-8659.00413.

[2] Chen Wang. "Kernel learning for visual perception". Capítulo 2.2.1, pp. 17–18. Tese de dout. Singapore: School of Electrical e Electronic Engineering, Nanyang Technological University, 2019. DOI: 10.32657/10220/47835. URL: https://hdl.handle.net/10220/47835.

[3] Liang Zhou, Chris R. Johnson e Daniel Weiskopf. "Data-Driven Space-Filling Curves". Em: *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2020), pp. 1591–1600. DOI: 10.1109/TVCG.2020.3030473.