

# Algoritmo Seam Carving aplicado à Visão Computacional

**Equipe:** André Albuquerque & Lucas Pinheiro

**Disciplina:** Projeto e Análise de Algoritmos

# TEMAS DE MESTRADO

- Navegação interna para pessoas com deficiência visual
  - Uso de visão computacional para percepção do ambiente interno.
  - Identificação de obstáculos, bordas, portas e caminhos livres.
- Detecção de falhas em pás eólicas
  - Inspeção visual automatizada usando imagens de alta resolução.
  - Detecção de falhas visuais.



Fonte: OpenAI, (2025)



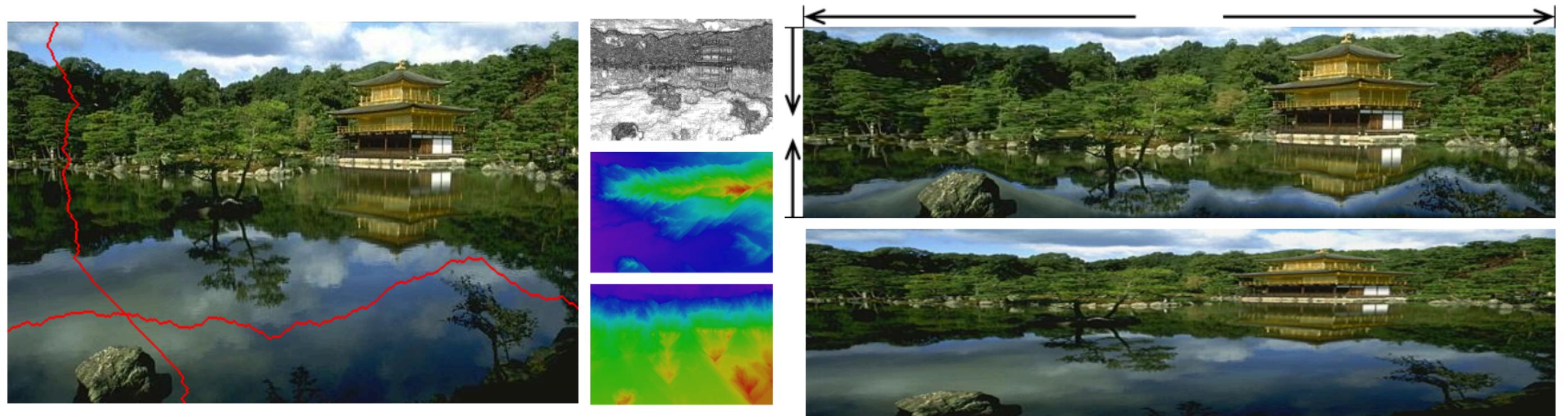
Fonte: Katsaprakakis, Papadakis e Ntintakis, (2021)

# CONTEXTUALIZANDO

- Problema geral:
  - Imagens de alta resolução.
  - Processamento computacionalmente custoso.
- Necessidade:
  - Reduzir dados visuais
  - Preservar informações importantes
  - Processamento em sistemas embarcados.
- **Seam Carving** ←

# O QUE É SEAM CARVING?

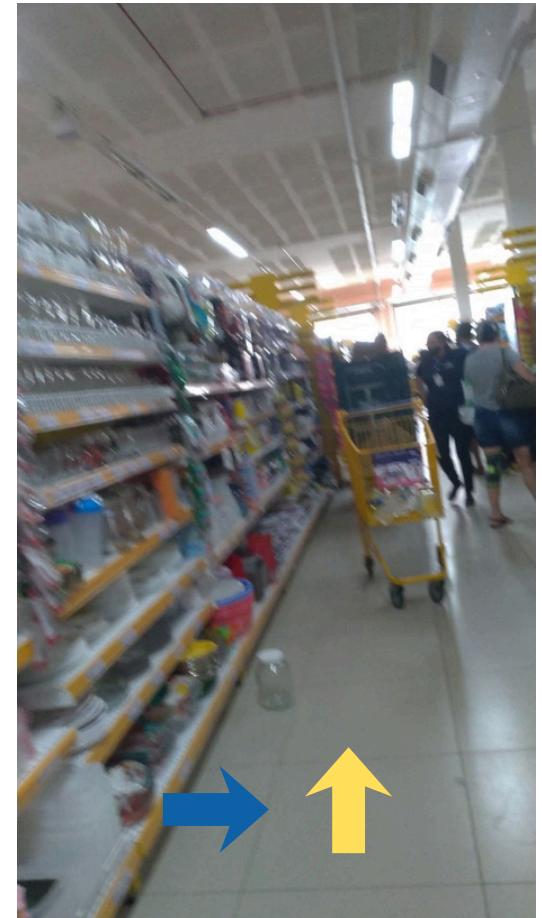
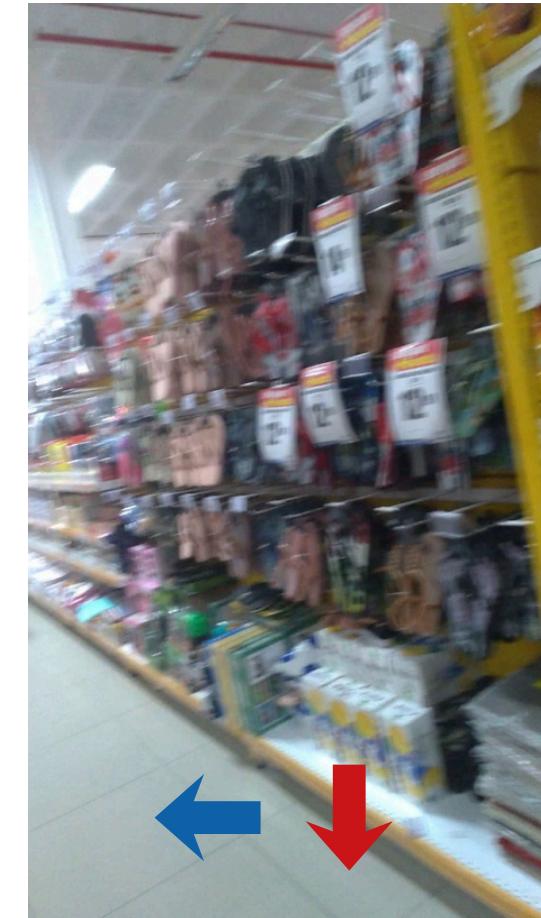
O Seam Carving é um algoritmo de redimensionamento sensível ao conteúdo, que remove ou insere caminhos contínuos de pixels com baixa importância visual (Avidan e Shamir, 2007).



**Fonte:** Avidan e Shamir, (2007)

# TEMA 1: AUXÍLIO A PESSOAS COM DEFICIÊNCIA VISUAL

- Adaptação de imagens para telas menores.
- Realce de regiões importantes:
  - objetos,
  - pessoas,
  - textos.
- O Seam Carving permite:
  - facilitar na interpretação de imagens.
  - preservar áreas semanticamente relevantes.



**Fonte:** Medeiros et. al, (2021)

# TEMA 2: DETECÇÃO DE FALHAS EM PÁS EÓLICAS

- Em inspeções visuais de pás eólicas:

- Trincas
  - Erosão
  - Danos a pintura



- O Seam Carving permite:

- preserva regiões de alta energia (bordas, trincas)
  - remove áreas homogêneas sem relevância estrutural



**Fonte:** Adaptado de Reddy et. al, (2019)

# FUNÇÃO DE ENERGIA

- A energia representa a importância visual de um pixel.

$$E(x, y) = \left| \frac{\partial I}{\partial x} \right| + \left| \frac{\partial I}{\partial y} \right|$$

- Bordas, contornos ou estruturas relevantes → alta energia
- Regiões homogêneas → baixa energia

# FORMULAÇÃO DO PROBLEMA (SEAM CARVING)

- Um seam vertical é um caminho de pixels.

Seam vertical

$$s = \{(x(i), i)\}_{i=1}^N \text{ com } |x(i) - x(i - 1)| \leq 1$$

- Contém exatamente um pixel por linha.
- Vai do topo à base da imagem.

Custo de um Seam

$$E(s) = \sum_{i=1}^N e(x(i), i)$$

Objetivo

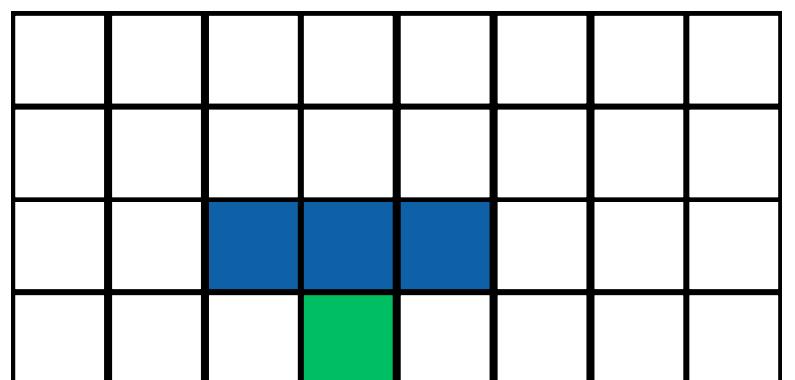
$$s^* = \arg \min_s \sum_{i=1}^N e(x(i), i)$$

# RESOLUÇÃO POR PROGRAMAÇÃO DINÂMICA

- O número de seams possíveis cresce **exponencialmente**.
- Avaliar todos os caminhos é **inviável**.
- O problema apresenta:
  - subproblemas sobrepostos.
  - estrutura de subproblema ótimo.
- A matriz M armazena o menor custo acumulado para cada pixel.
- O menor valor na última linha de M indica o final do seam ótimo.
- O caminho é recuperado por backtracking.

Seja  $M(i,j)$  o menor custo para alcançar o pixel  $(i,j)$

$$M(i, j) = e(i, j) + \min \begin{cases} M(i - 1, j - 1) \\ M(i - 1, j) \\ M(i - 1, j + 1) \end{cases}$$



# **FUNCIONAMENTO DO ALGORITMO**

**O algoritmo funciona em quatro etapas principais:**

1. Cálculo do mapa de energia.
2. Construção do mapa de energia acumulada.
3. Identificação do seam mínimo.
4. Remoção ou inserção do seam.

# Análise de Complexidade do Algoritmo

```
1 ALGORITMO SeamCarvingVertical(imagem I):
2
3     E <- CalcularEnergia(I)
4     criar matriz M de tamanho N x M
5     criar matriz P de tamanho N x M
6
7     PARA j de 0 até M-1:
8         M[0][j] <- E[0][j]
9         P[0][j] <- -1
10
11    PARA i de 1 até N-1:
12        PARA j de 0 até M-1:
13            M[i][j] <- E[i][j] + min(
14                M[i-1][j-1], M[i-1][j], M[i-1][j+1])
15            P[i][j] <- coluna correspondente ao menor valor
16
17    j_min <- coluna j que minimiza M[N-1][j]
18
19    PARA i de N-1 até 0:
20        seam[i] <- j_min
21        j_min <- P[i][j_min]
22
23    remover, em cada linha i, o pixel da coluna seam[i]
24
25    RETORNAR imagem reduzida
```

```

1 ALGORITMO SeamCarvingVertical(imagem I):
2
3     E <- CalcularEnergia(I)           → Θ(NxM)
4     criar matriz M de tamanho N x M   → Θ(NxM)
5     criar matriz P de tamanho N x M   → Θ(NxM)
6
7     PARA j de 0 até M-1:
8         M[0][j] <- E[0][j]
9         P[0][j] <- -1
10
11    PARA i de 1 até N-1:
12        PARA j de 0 até M-1:
13            M[i][j] <- E[i][j] + min(
14                M[i-1][j-1], M[i-1][j], M[i-1][j+1])
15            P[i][j] <- coluna correspondente ao menor valor
16
17    j_min <- coluna j que minimiza M[N-1][j]
18
19    PARA i de N-1 até 0:
20        seam[i] <- j_min
21        j_min <- P[i][j_min]
22
23    remover, em cada linha i, o pixel da coluna seam[i]
24
25    RETORNAR imagem reduzida

```

```

1 ALGORITMO SeamCarvingVertical(imagem I):
2
3     E <- CalcularEnergia(I)           → Θ(NxM)
4     criar matriz M de tamanho N x M   → Θ(NxM)
5     criar matriz P de tamanho N x M   → Θ(NxM) } Θ(NxM)
6
7     PARA j de 0 até M-1:
8         M[0][j] <- E[0][j]
9         P[0][j] <- -1
10
11    PARA i de 1 até N-1:
12        PARA j de 0 até M-1:
13            M[i][j] <- E[i][j] + min(
14                M[i-1][j-1], M[i-1][j], M[i-1][j+1])
15            P[i][j] <- coluna correspondente ao menor valor
16
17    j_min <- coluna j que minimiza M[N-1][j]
18
19    PARA i de N-1 até 0:
20        seam[i] <- j_min
21        j_min <- P[i][j_min]
22
23    remover, em cada linha i, o pixel da coluna seam[i]
24
25    RETORNAR imagem reduzida

```

```

1 ALGORITMO SeamCarvingVertical(imagem I):
2
3     E <- CalcularEnergia(I)           → Θ(NxM)
4     criar matriz M de tamanho N x M   → Θ(NxM)
5     criar matriz P de tamanho N x M   → Θ(NxM) } Θ(NxM)
6
7     PARA j de 0 até M-1:             → M vezes → Θ(M)
8         M[0][j] <- E[0][j]           → Θ(1)
9         P[0][j] <- -1              → Θ(1)
10
11    PARA i de 1 até N-1:
12        PARA j de 0 até M-1:
13            M[i][j] <- E[i][j] + min(
14                M[i-1][j-1], M[i-1][j], M[i-1][j+1])
15            P[i][j] <- coluna correspondente ao menor valor
16
17    j_min <- coluna j que minimiza M[N-1][j]
18
19    PARA i de N-1 até 0:
20        seam[i] <- j_min
21        j_min <- P[i][j_min]
22
23    remover, em cada linha i, o pixel da coluna seam[i]
24
25    RETORNAR imagem reduzida

```

```

1 ALGORITMO SeamCarvingVertical(imagem I):
2
3     E <- CalcularEnergia(I)           → Θ(NxM)
4     criar matriz M de tamanho N x M   → Θ(NxM)
5     criar matriz P de tamanho N x M   → Θ(NxM) } Θ(NxM)
6
7     PARA j de 0 até M-1:             → M vezes → Θ(M)
8         M[0][j] <- E[0][j]          → Θ(1)
9         P[0][j] <- -1              → Θ(1)
10
11    PARA i de 1 até N-1:            → N - 1 vezes
12        PARA j de 0 até M-1:        → M vezes
13            M[i][j] <- E[i][j] + min(
14                M[i-1][j-1], M[i-1][j], M[i-1][j+1])
15            P[i][j] <- coluna correspondente ao menor valor } Θ(NxM)
16
17    j_min <- coluna j que minimiza M[N-1][j]
18
19    PARA i de N-1 até 0:
20        seam[i] <- j_min
21        j_min <- P[i][j_min]
22
23    remover, em cada linha i, o pixel da coluna seam[i]
24
25    RETORNAR imagem reduzida

```

```

1 ALGORITMO SeamCarvingVertical(imagem I):
2
3     E <- CalcularEnergia(I)           → Θ(NxM)
4     criar matriz M de tamanho N x M   → Θ(NxM)
5     criar matriz P de tamanho N x M   → Θ(NxM) } Θ(NxM)
6
7     PARA j de 0 até M-1:             → M vezes → Θ(M)
8         M[0][j] <- E[0][j]           → Θ(1)
9         P[0][j] <- -1              → Θ(1)
10
11    PARA i de 1 até N-1:            → N - 1 vezes
12        PARA j de 0 até M-1:        → M vezes
13            M[i][j] <- E[i][j] + min(
14                M[i-1][j-1], M[i-1][j], M[i-1][j+1])
15            P[i][j] <- coluna correspondente ao menor valor } Θ(NxM)
16
17    j_min <- coluna j que minimiza M[N-1][j]
18
19    PARA i de N-1 até 0:           → N vezes → Θ(N)
20        seam[i] <- j_min          → Θ(1)
21        j_min <- P[i][j_min]       → Θ(1)
22
23    remover, em cada linha i, o pixel da coluna seam[i] → Θ(N)
24
25    RETORNAR imagem reduzida

```

```

1 ALGORITMO SeamCarvingVertical(imagem I):
2
3     E <- CalcularEnergia(I)           → Θ(NxM)
4     criar matriz M de tamanho N x M   → Θ(NxM)
5     criar matriz P de tamanho N x M   → Θ(NxM) } Θ(NxM)
6
7     PARA j de 0 até M-1:             → M vezes → Θ(M)
8         M[0][j] <- E[0][j]           → Θ(1)
9         P[0][j] <- -1              → Θ(1)
10
11    PARA i de 1 até N-1:            → N - 1 vezes
12        PARA j de 0 até M-1:        → M vezes
13            M[i][j] <- E[i][j] + min(
14                M[i-1][j-1], M[i-1][j], M[i-1][j+1])
15            P[i][j] <- coluna correspondente ao menor valor } Θ(NxM)
16
17    j_min <- coluna j que minimiza M[N-1][j]
18
19    PARA i de N-1 até 0:           → N vezes → Θ(N)
20        seam[i] <- j_min          → Θ(1)
21        j_min <- P[i][j_min]       → Θ(1)
22
23    remover, em cada linha i, o pixel da coluna seam[i] → Θ(N)
24
25    RETORNAR imagem reduzida

```

**Custo Total:**

$$\Theta(N \cdot M)$$

# CONCLUSÃO

# CONCLUSÃO

- Complexidade  $\Theta(N \cdot M)$  no pior caso.
- O Seam Carving pode ser integrado como:
  - etapa de pré-processamento
  - mecanismo de redução seletiva de informação visual
- A mesma base algorítmica pode ser aplicada a:
  - navegação interna para pessoas com deficiência visual
  - detecção de rachaduras em pás eólicas