Processamento Digital de Imagens

Prof. Bogdan Tomoyuki Nassu

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Desafio

- *Inpainting*.
  - Reconstruir partes perdidas ou deterioradas de uma imagem ou pintura.
  - Ex: restauração de pinturas (realizada por artista).

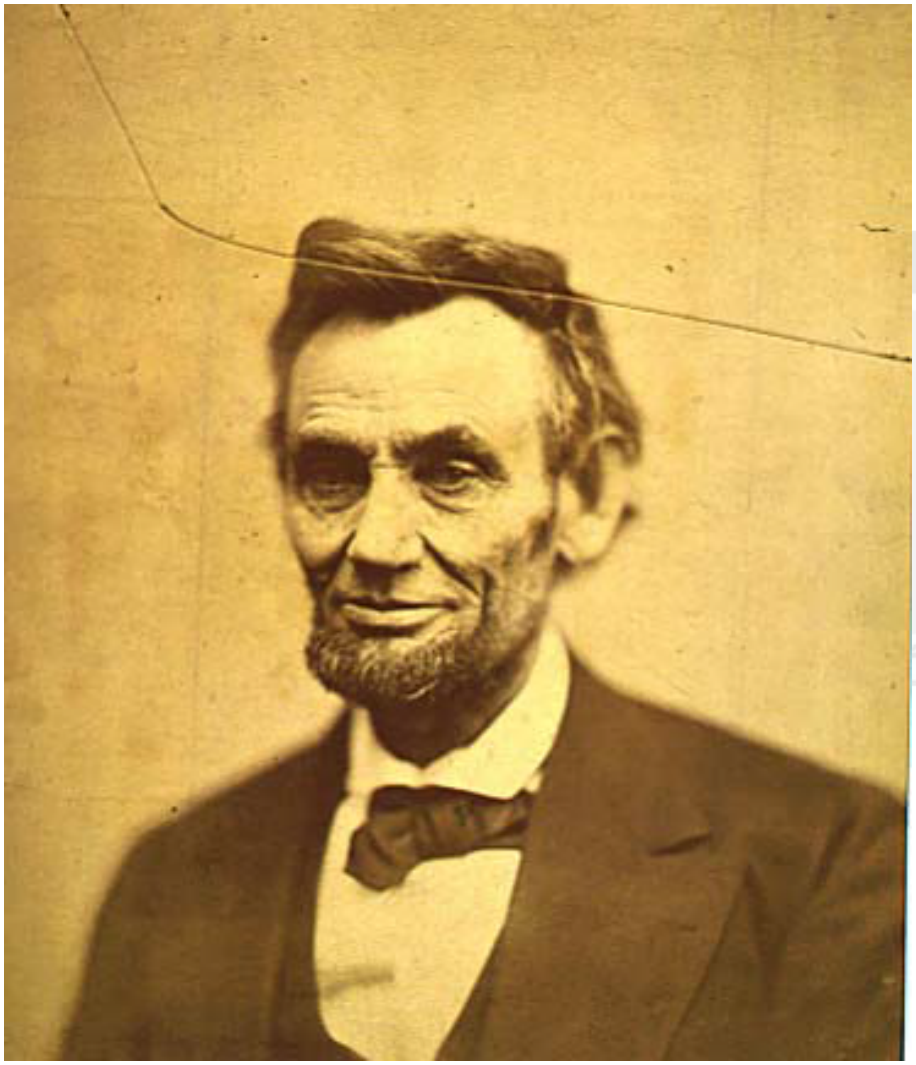
Desafio

- *Inpainting*.
  - Reconstruir partes perdidas ou deterioradas de uma imagem ou pintura.
  - Ex: restauração de pinturas (realizada por artista competente).

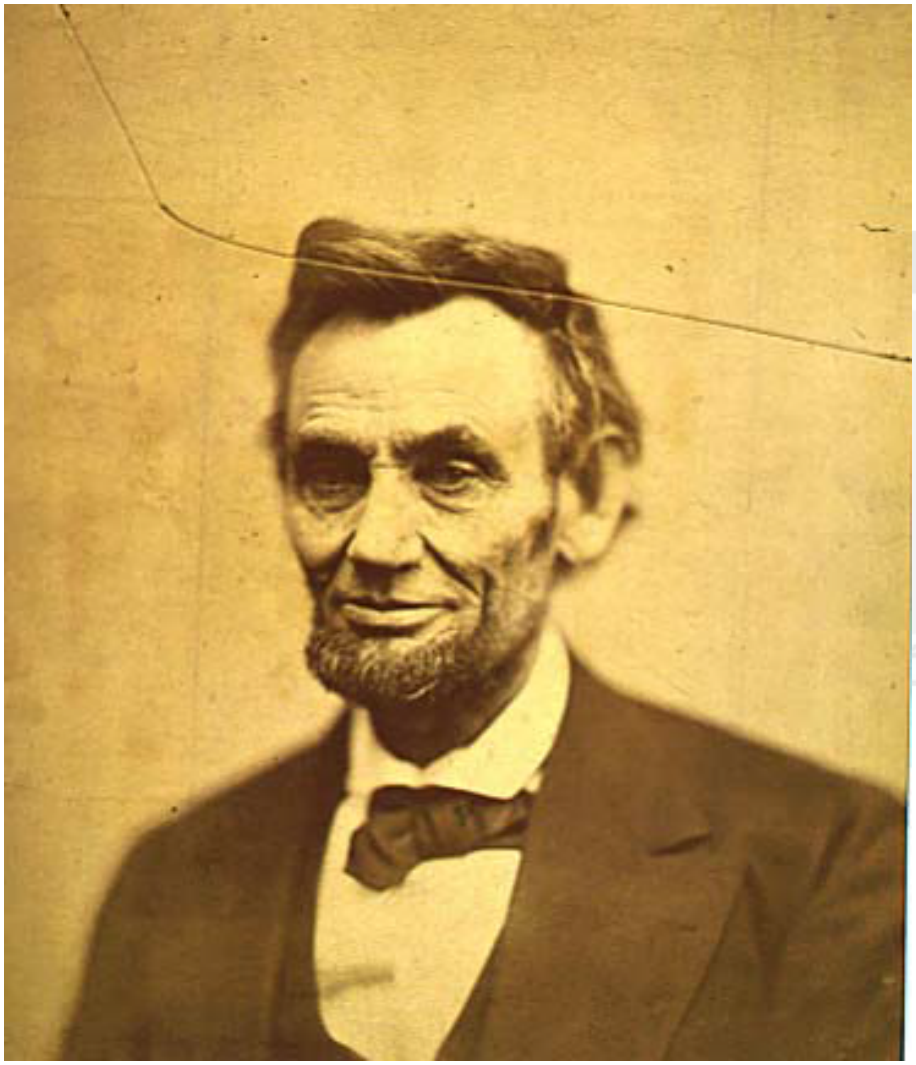
Inpainting por algoritmos

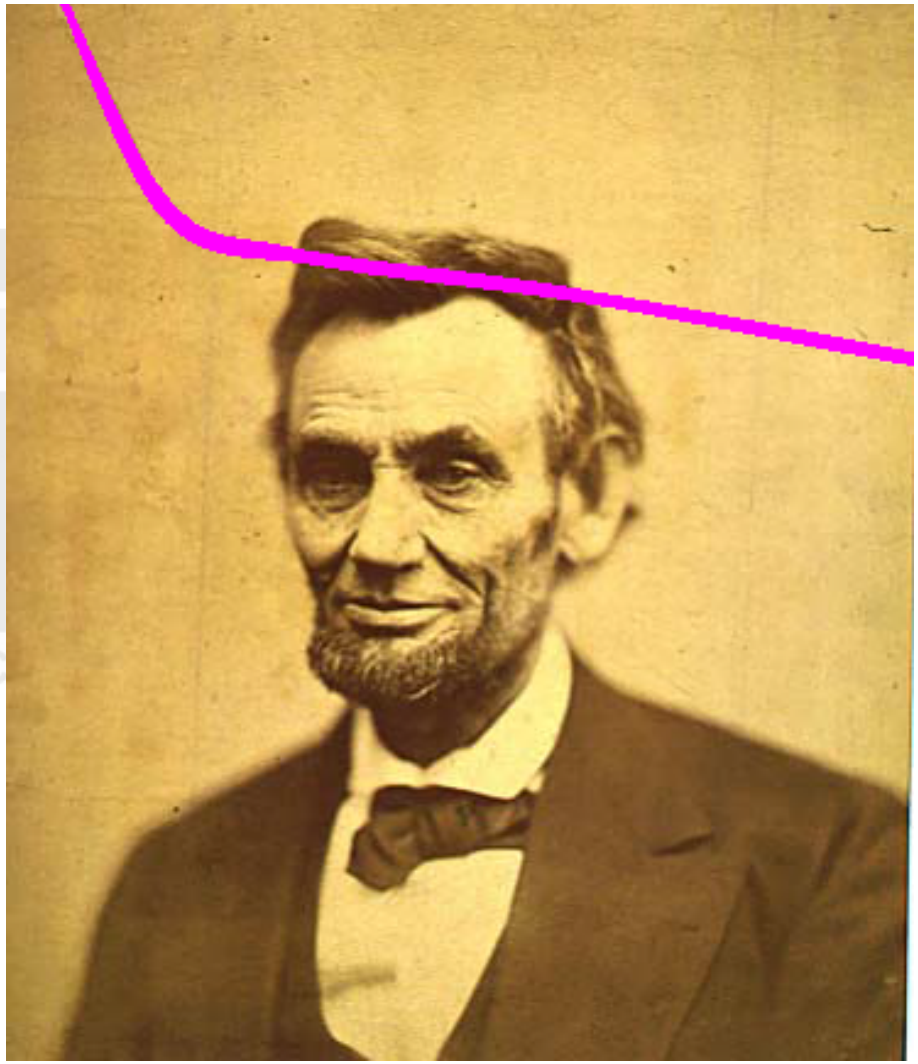
- Algoritmos podem (tentar) realizar a restauração automaticamente.
  - Objetivo: preencher ou substituir uma região da imagem.
  - A região a se preencher normalmente é informada manualmente.
    - Esta região é uma “máscara para substituição”.
- Exemplos de aplicações:
  - Restauração de filmes antigos digitalizados.
  - Remoção de objetos de fotografias.
  - Completar blocos perdidos em transmissões de vídeo.

Inpainting por algoritmos

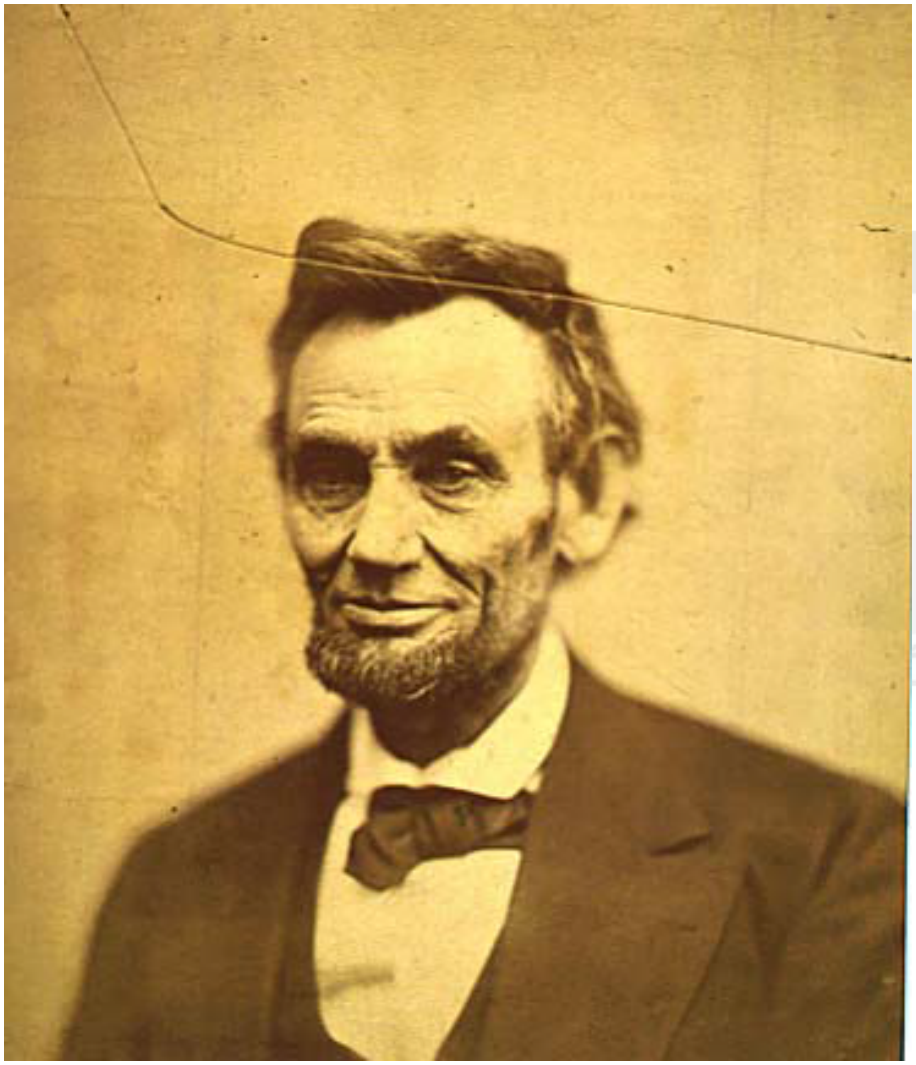


Inpainting por algoritmos

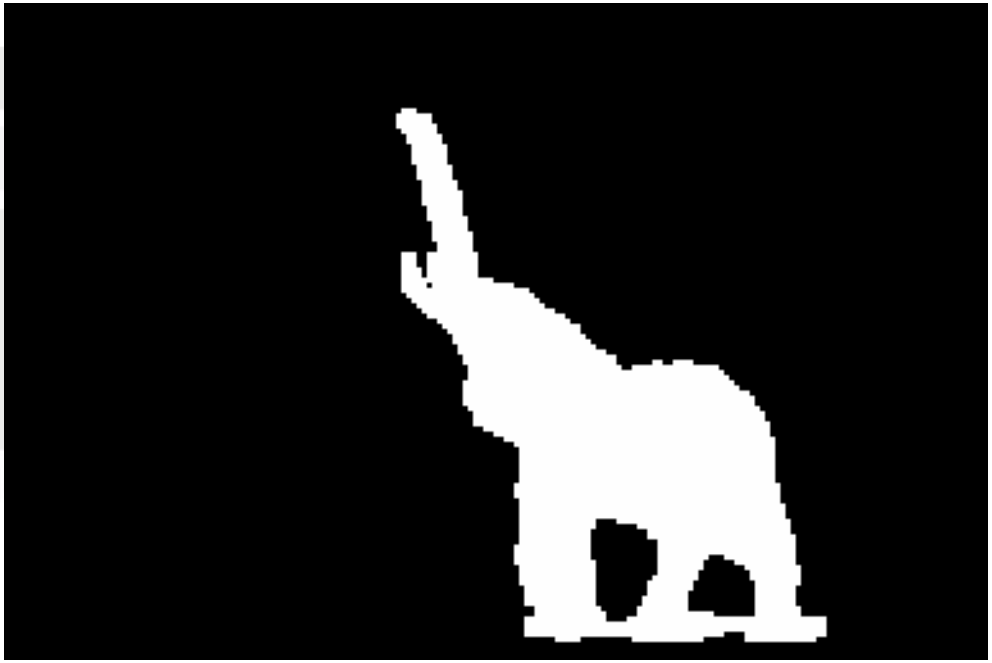




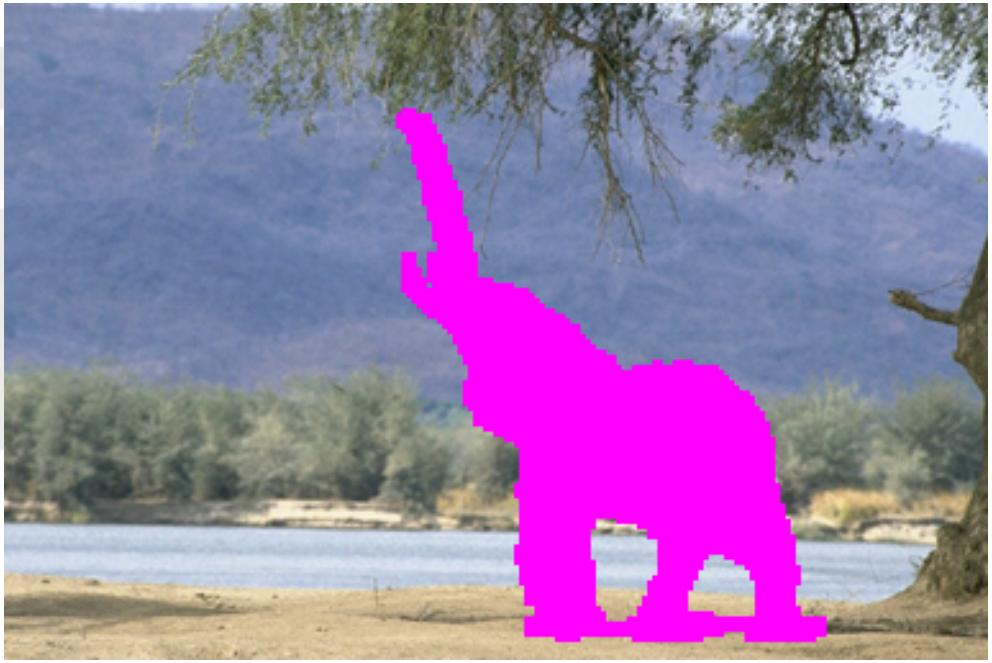
Inpainting por algoritmos



Inpainting por algoritmos



Inpainting por algoritmos



Inpainting por algoritmos

Como fazer?

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Como fazer?

- Algumas formas de pensar:
  - A região é preenchida como uma continuação dos seus arredores.
  - A região é preenchida com outras regiões da imagem.
  - A região é preenchida com regiões de outras imagens.
  - A região é preenchida com a mesma região em outros frames (vídeo).
  - A região é preenchida com texturas geradas pelo próprio algoritmo.
  - (Uma mistura de ideias?)

Hoje

- Hoje, testaremos 3 ideias diferentes:
  - Um algoritmo que “vaza” dados da vizinhança para dentro da região.
  - Um algoritmo que preenche a região com outras partes da imagem.
  - Um algoritmo que gera valores para os pixels.

Ideia #1: “vazando a vizinhança”

•Nossa primeira tentativa envolve “vazar” dados da vizinhança para dentro da região.

- Como isso pode ser feito?

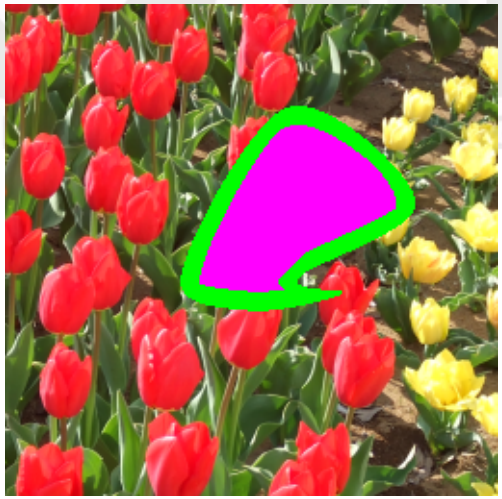
UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Sequência de preenchimento

- A região é preenchida de fora para dentro.
  - Como garantir esta sequência de preenchimento?

Magenta: máscara para substituição (a preencher).
Verde: regiões já preenchidas.

Máscara inicial.



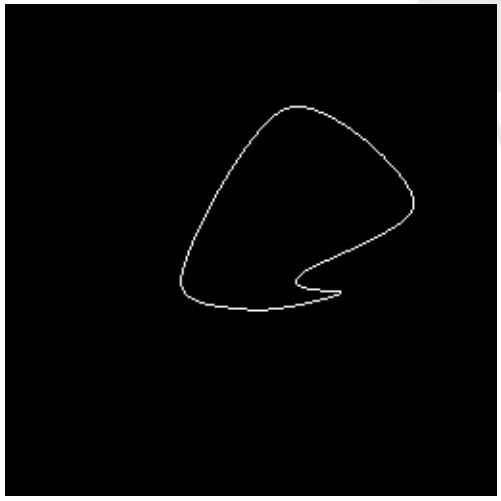
Após 10 iterações.

Após 20 iterações.

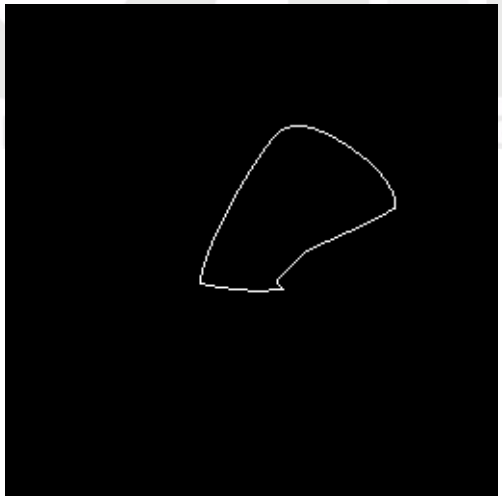
Após 30 iterações.

Sequência de preenchimento

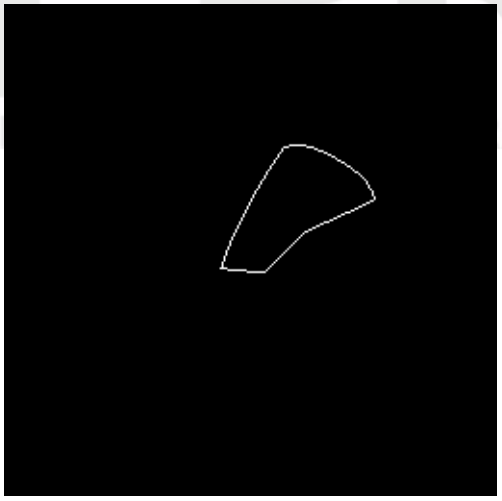
```
enquanto existirem pixels a se substituir:
    encontra o contorno interno da máscara:
        erodida = erode máscara (usa um kernel de vizinhança-4)
        contorno = máscara - erodida
    substitui cada pixel do contorno
    máscara = erodida
```



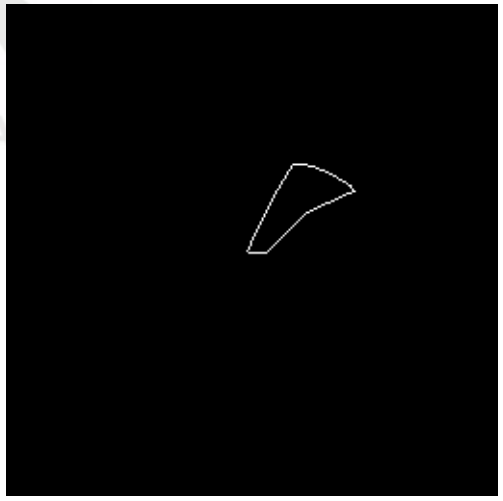
Primeira iteração.



Após 10 iterações.



Após 20 iterações.



Após 30 iterações.

Preencher com... ?

- Os pixels marcados para substituição são preenchidos com...?

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

•Os pixels marcados para substituição são preenchidos com uma média ponderada dos pixels válidos na sua vizinhança.

- Pixel válido = pixel que não está marcado na máscara para substituição.

Para imagens coloridas, mantemos uma soma para cada canal.

```
substitui cada pixel do contorno:
    soma_pesos = 0
    soma = 0
    para os vizinhos acima, abaixo, à esquerda e à direita:
        se o vizinho estiver dentro da imagem e não estiver marcado
        na máscara para substituição:
            soma_pesos += 2
            soma += valor do vizinho * 2
    para os vizinhos nas diagonais:
        se o vizinho estiver dentro da imagem e não estiver marcado
        na máscara para substituição:
            soma_pesos++
            soma += valor do vizinho

    valor do pixel = soma / soma_pesos
```

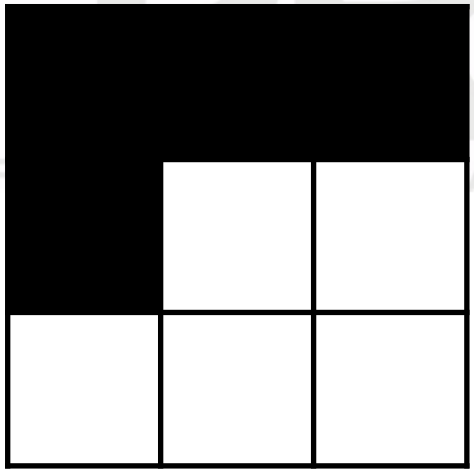
UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Exemplo

- No exemplo abaixo, o pixel será substituído por:
  - (0.3 * 1 + 0.1 * 2 + 0.2 * 1 + 0.6 * 2) / 6 = 0.3167
- Como será a aparência da imagem nas regiões substituídas?

0.3	0.1	0.2
0.6	?	?
?	?	?

Vizinhança
(imagem atual)



Vizinhança
(máscara)

Máscara inicial.

Após 6 iterações.

Após 13 iterações.

Após 19 iterações.

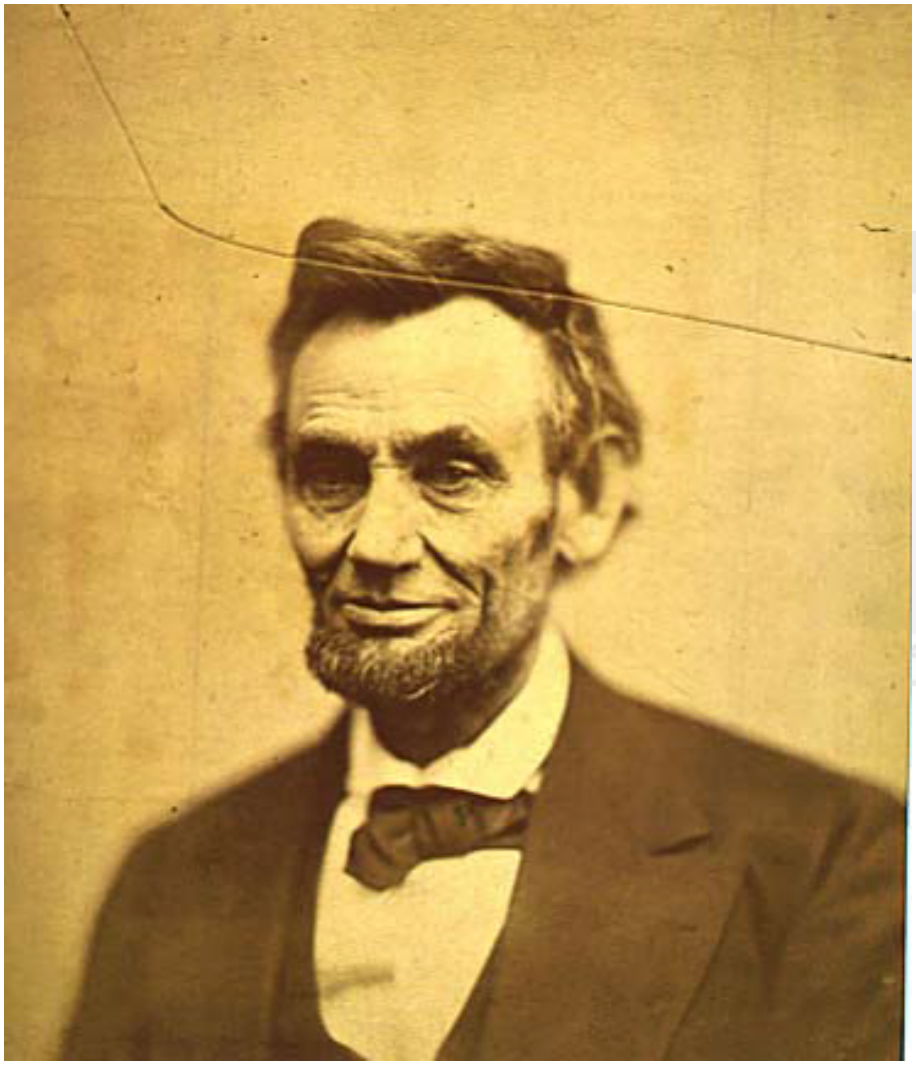
Após 25 iterações.

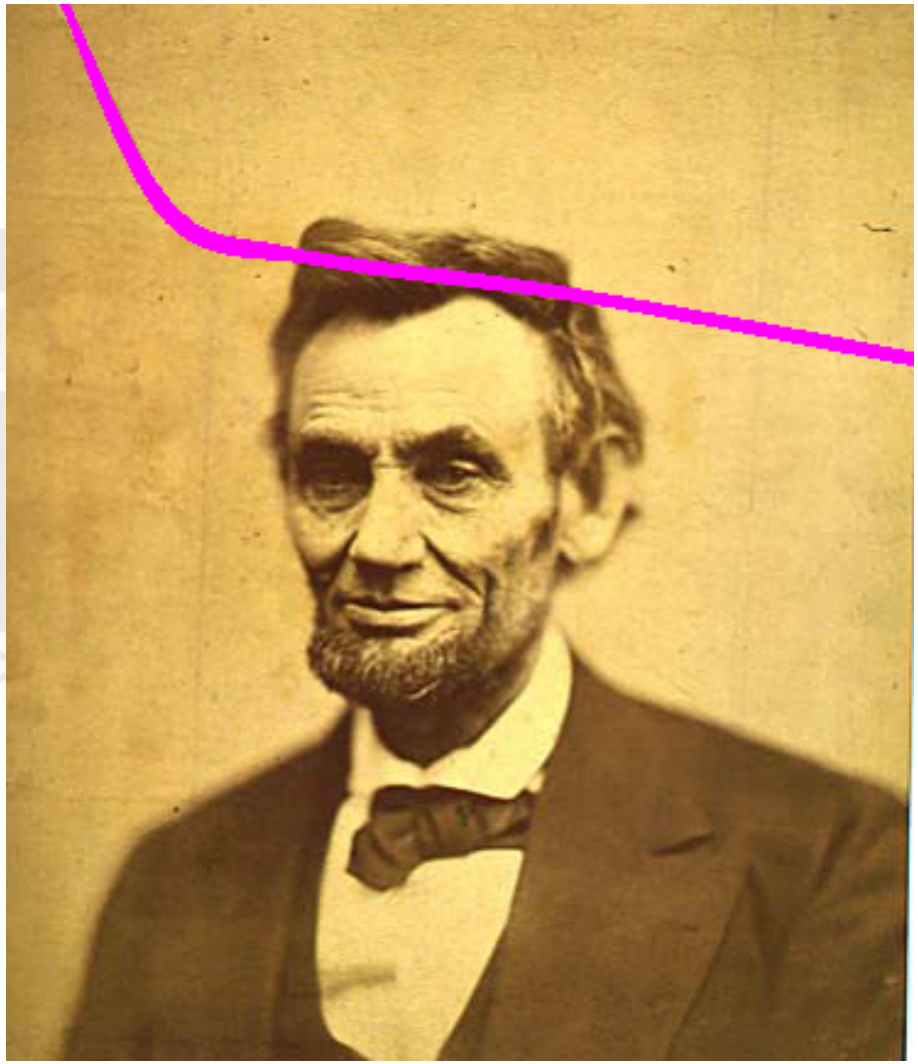
Após 31 iterações.

Após 38 iterações.

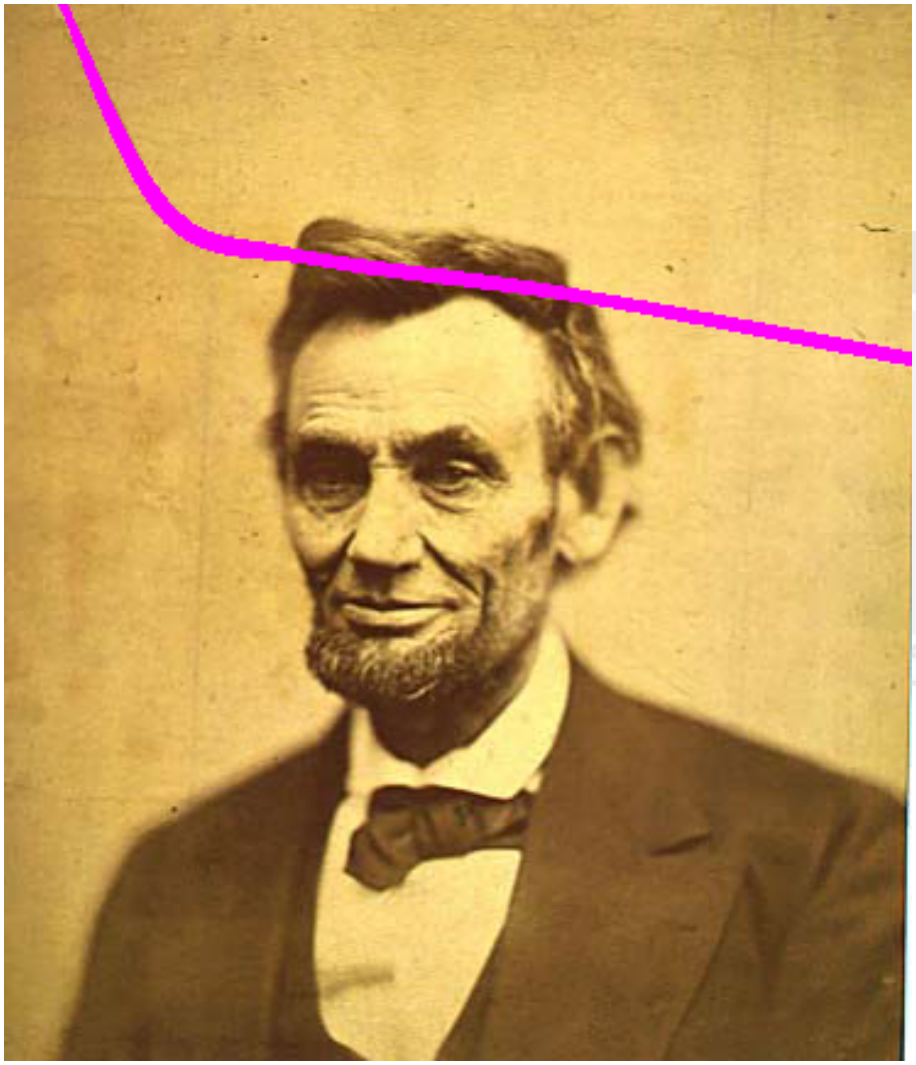
Após 44 iterações.

Exemplo

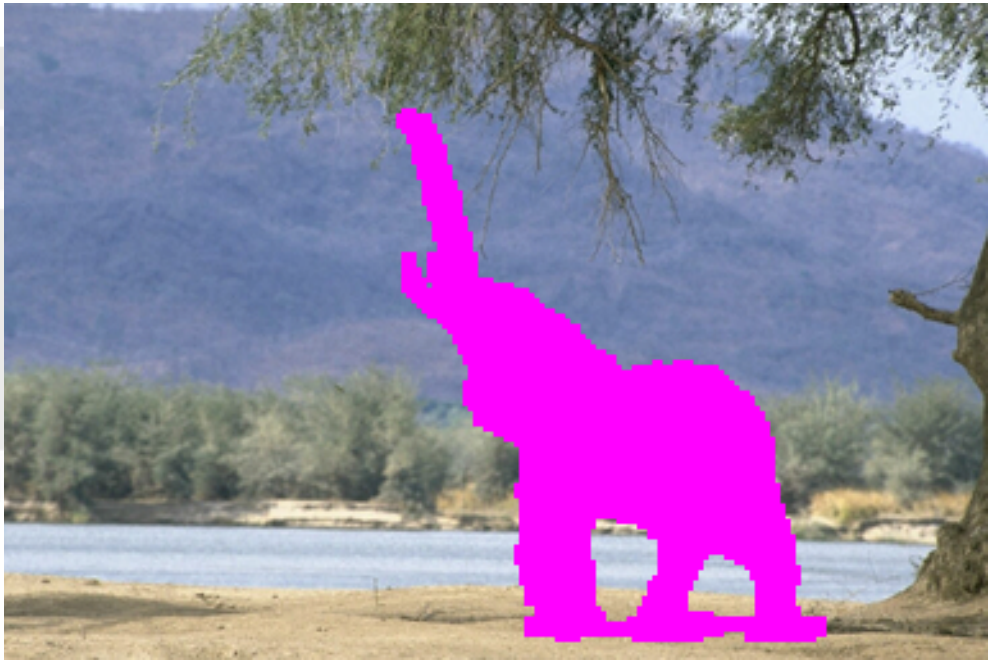




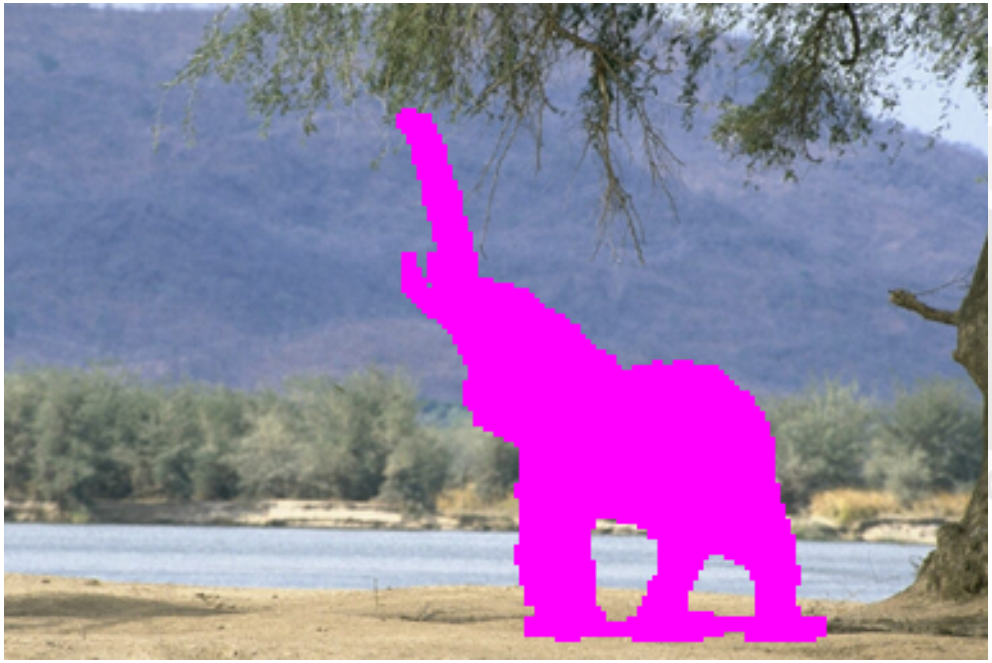
Exemplo



Exemplo



Exemplo



Ideia #1: comentários

- Nesta abordagem, os dados são escolhidos seguindo um padrão parecido com o kernel de um filtro Gaussiano 3x3.
  - Poderíamos usar outros padrões, com vizinhanças maiores ou pesos diferentes.
    - Ex: vizinhanças maiores → mais borramento, “costura” no centro menos visível.

- O OpenCV tem funções para *inpainting* que seguem este princípio.
  - Telea, que considera a homogeneidade e os gradientes na vizinhança para decidir a ponderação da vizinhança tentando prolongar estruturas.
  - Navier-Stokes, que modela o problema usando dinâmica dos fluidos (!) - a modelagem é muito diferente, mas os resultados são parecidos.

Telea

Abordagem simples

Telea

Navier-Stokes

Abordagem simples

Navier-Stokes

Ideia #2: “pegando da imagem”

•Nossa segunda tentativa envolve substituir as regiões marcadas na máscara por outras partes da imagem.

- Como isso pode ser feito?
  - = como escolher as regiões que substituirão os “buracos”?

Ideia #2: “pegando da imagem”

- Podemos selecionar uma vizinhança em volta da região a ser substituída, e procurar na imagem uma vizinhança parecida.
  - Pensamento: se encontrarmos uma vizinhança parecida, o “miolo” (desconhecido) também pode ser parecido.

Região a substituir.

Acha uma vizinhança parecida...

Copia o “miolo”.

Como determinar a vizinhança?

- = Quais pixels serão comparados com outras regiões da imagem?
- Como?

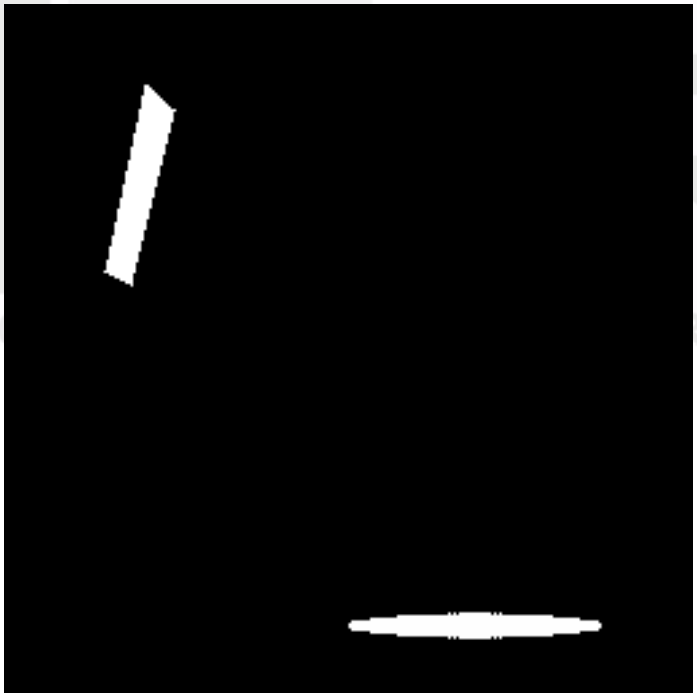
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Como determinar a vizinhança?

•Basta dilatar a máscara com um kernel redondo grande, e obter a diferença entre a imagem dilatada e a máscara original.

- Importante: cada componente conexo da máscara deve ser analisado separadamente!

Imagem.



Máscara para substituição.

Vizinhança (blob 1).

Vizinhança (blob 2).

Vizinhança parecida

- Como podemos encontrar uma vizinhança parecida?

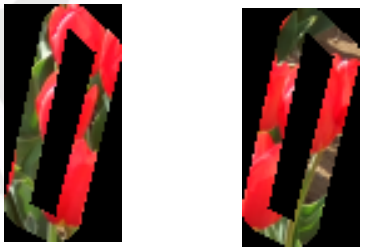
UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Vizinhança parecida

- Para cada componente conexo, usamos uma janela deslizante:
  - Geramos uma pequena máscara indicando o formato da vizinhança.
  - Pegamos os pixels da vizinhança na posição original.
  - Posicionamos a máscara em diferentes locais da imagem, e comparamos as vizinhanças.

Vizinhança original.

Vizinhança testada (janela deslizante).



Compara as vizinhanças. Escolhe a mais parecida com a original.

Comparando vizinhanças

- Como comparar as duas vizinhanças?

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Comparando vizinhanças

- Uma medida simples:
  - *m*: máscara com o formato da vizinhança.
  - *w*: largura de *m*.
  - *h*: altura de *m*.
  - *f*: imagem de entrada.
  - (x,y): canto superior esquerdo da vizinhança testada.
  - (x_0,y_0): canto superior esquerdo da vizinhança original.

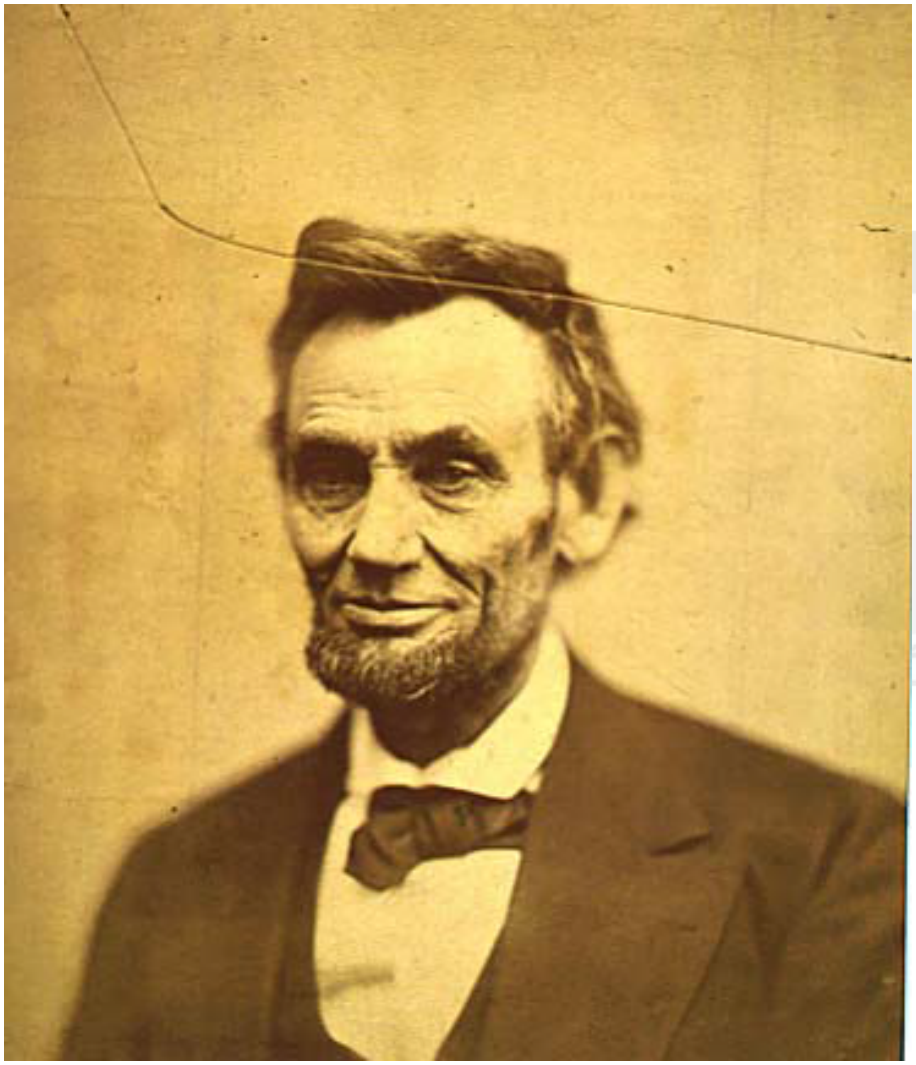
```
score = 0
for (i = 0; i < h; i++)
    for (j = 0; j < w; j++)
        if (m(j,i) está setada)
            score += |f(x+j,y+i) - f(x0+j,y0+i)|
```

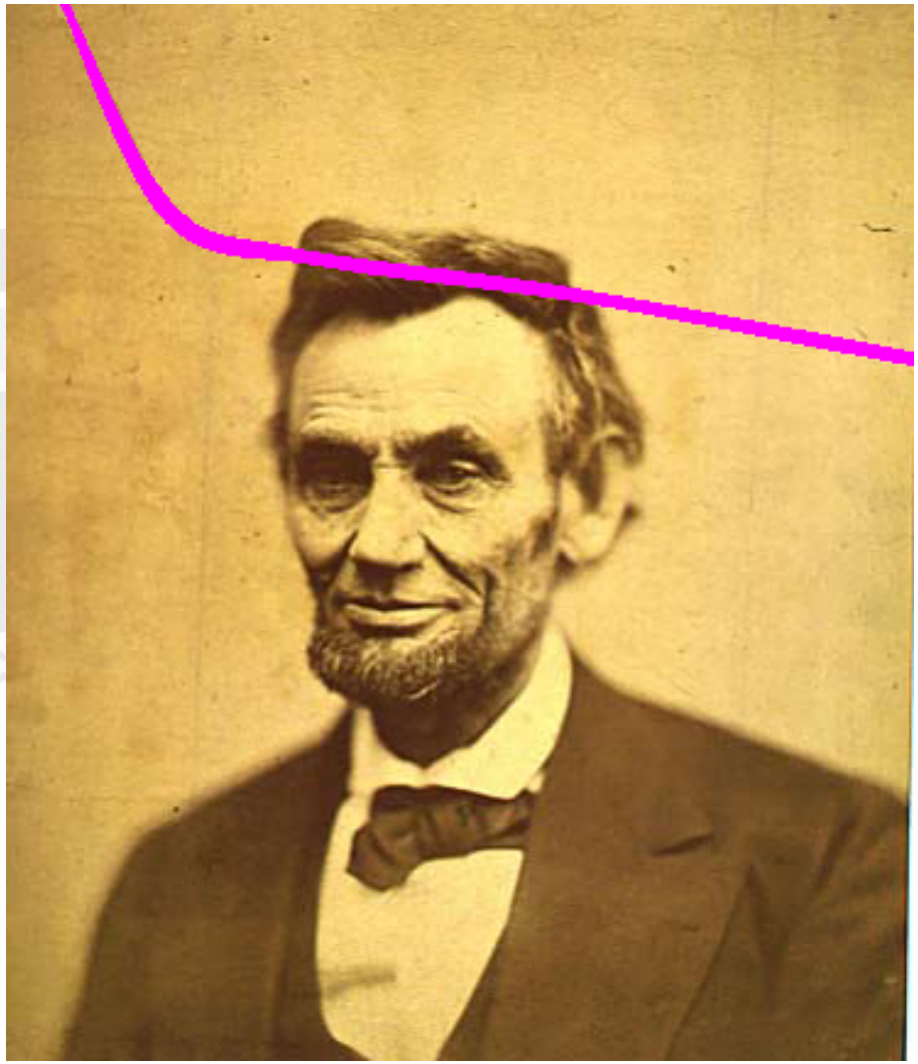
Para imagens coloridas, o score é atualizado para cada canal.

Nota: uma posição (x,y) é inválida se existir algum pixel na janela deslizante setado na máscara de substituição e que precise ser usado para calcular o score ou para a substituição final.

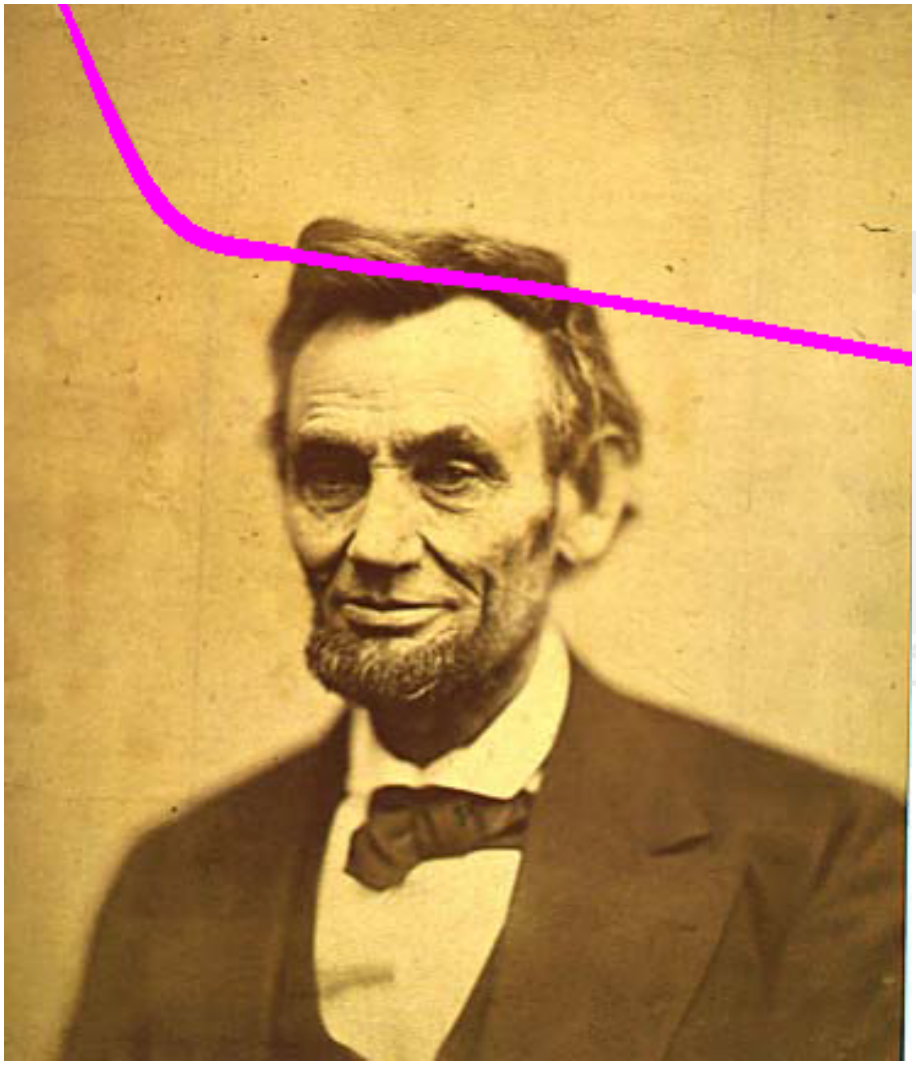
Exemplo

Exemplo

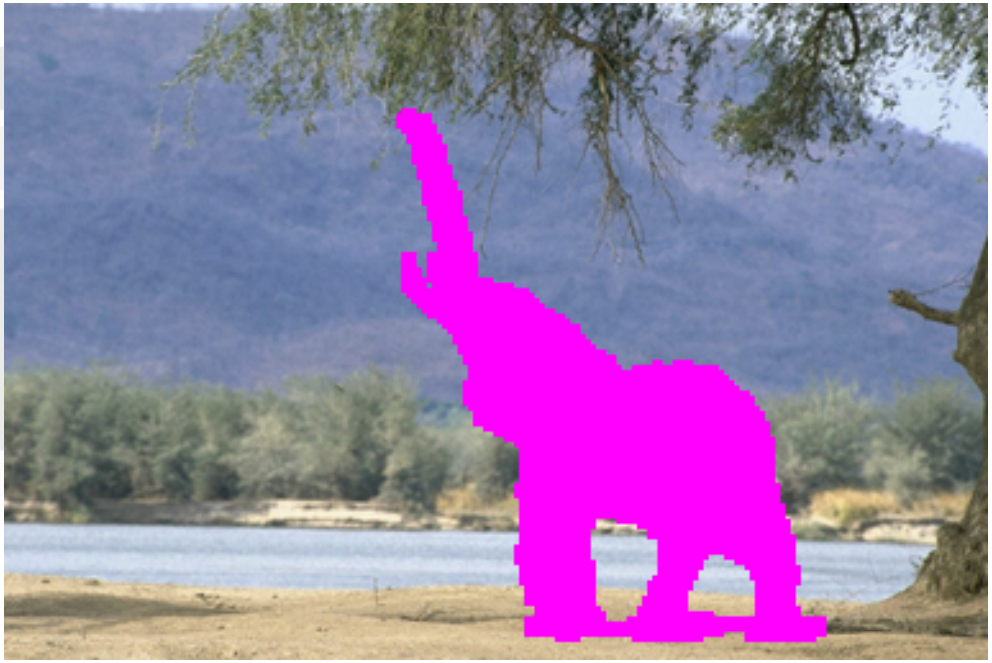




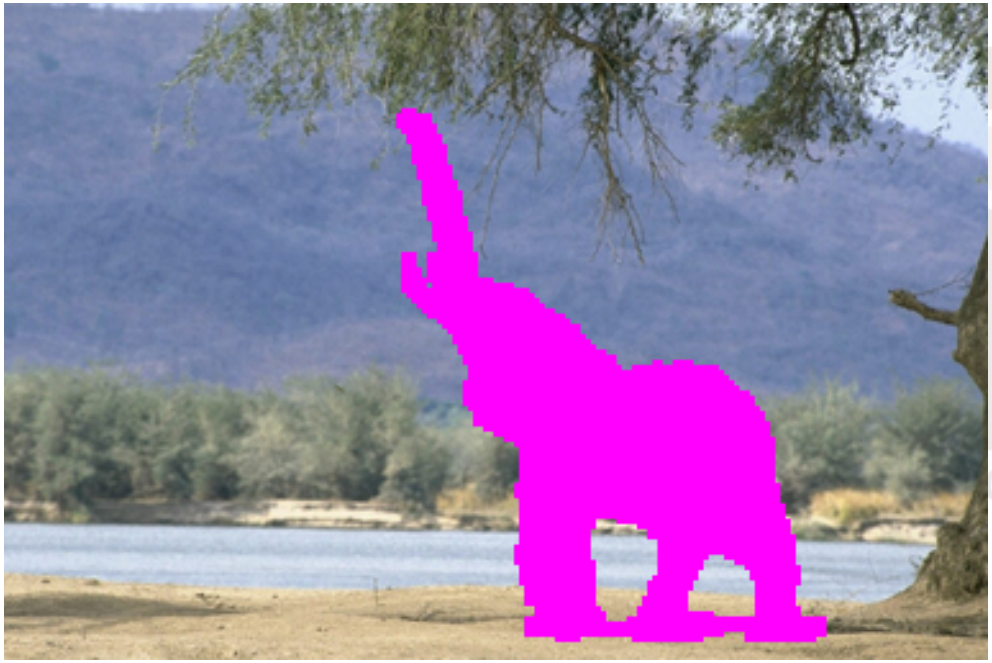
Exemplo



Exemplo



Exemplo

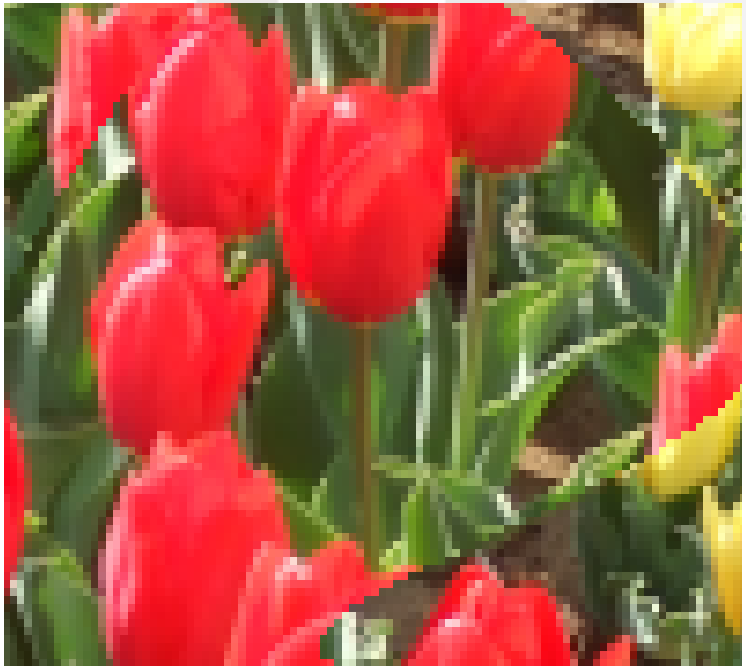


Ideia #2: comentários

•Uma variação desta ideia que costuma funcionar bem é realizar a busca em outros quadros de um vídeo.

•A abordagem pode ser modificada para suavizar a fronteira entre a região substituída e a vizinhança (*alpha blending*).

- A vizinhança e o número de posições inválidas ficam maiores.



Ideia #3: gerando texturas

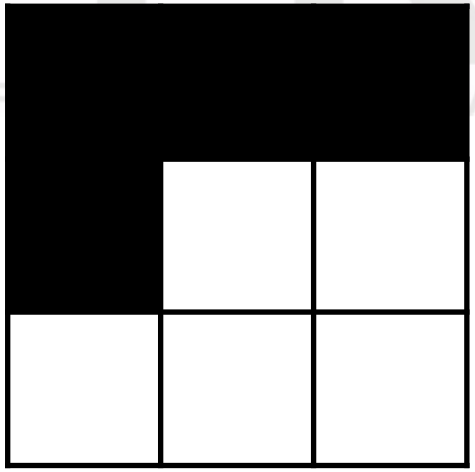
- Imagine uma variação da ideia anterior que trabalha com pixels individuais, em vez da região completa.
  - = para cada pixel na máscara de substituição, toma uma vizinhança e procura por outra vizinhança similar na imagem.
- Como os pixels serão substituídos um a um, usaremos pixels da imagem original para gerar novos padrões de textura!

Exemplo

- No exemplo abaixo, usamos uma vizinhança 3x3.
  - Usando uma janela deslizante, procuramos na imagem original outra vizinhança parecida.
  - Podemos usar a mesma medida de pontuação da ideia anterior.
  - O score é computado usando apenas os pixels que não estão setados na máscara (neste exemplo, p1, p2, p3 e p4).
  - Vizinhanças candidatas com pixels fora da imagem ou setados na máscara são inválidas.

p1	p2	p3
p4	p5	p6
p7	p8	p9

Vizinhança (imagem)



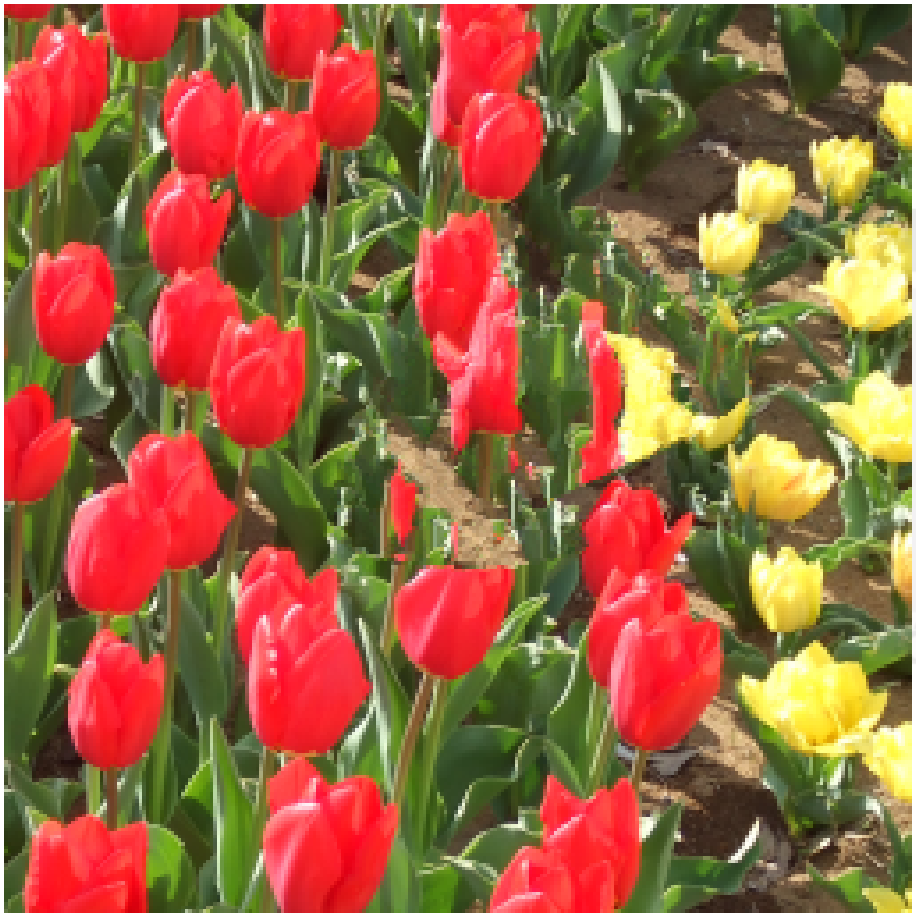
Vizinhança (máscara)

Ideia básica

```
for (cada pixel (x,y) tal que mask (x,y) ≠ 0)
   for (cada posição (x',y') da imagem)
      score = 0
      for (i = -w, i <= +w; i++) // w é a largura da janela
         for (j = -w; j <= +w; j++)
            se (mask(x+j,y+i) ≠ 0 ou (x+j,y+i) está fora da imagem)
               pula este (j,i) - este pixel não entra no score
            se (mask(x'+j,y'+i)≠0 ou (x'+j,y'+i) está fora da imagem)
               pula este (x',y') - esta posição é inválida
            score += |f(x+j,y+i) - f(x'+j,y'+i)|

   substitui o pixel (x,y) pelo pixel (x',y') que tiver o menor score
```

Exemplo



Testando vizinhança 3x3

Testando vizinhança 5x5

Exemplo

Testando vizinhança 9x9

Testando vizinhança 13x13

Sobre a ordem de substituição

- Substituir os pixels seguindo de cima para baixo e da esquerda para a direita produz transições mais suaves no topo / esquerda da região substituída, e mais agressivas abaixo / à direita.
- Podemos substituir de fora para dentro.
  - Funcionamento similar ao da primeira abordagem:

```
enquanto existirem pixels a se substituir:
    encontra o contorno interno da máscara
    substitui cada pixel do contorno
        (como discutido alguns slides atrás)
    remove da máscara os pixels substituídos na imagem
```

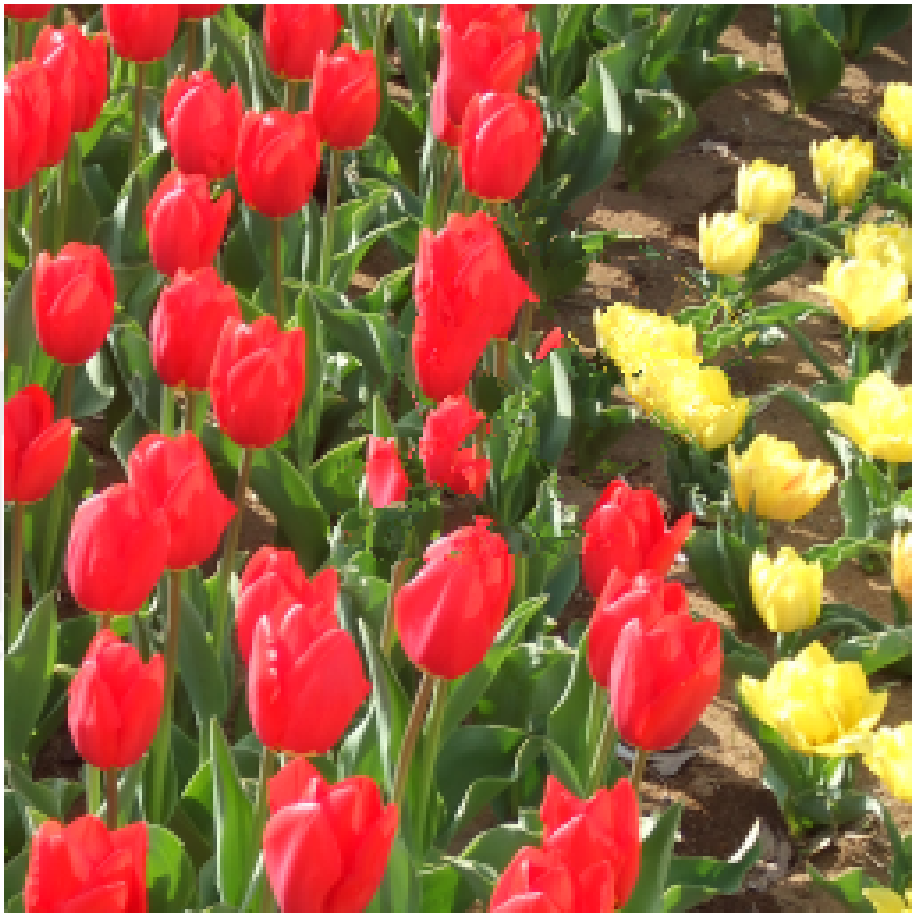
Exemplo

Testando vizinhança 5x5
(ordem de visita)

Testando vizinhança 5x5
(de fora para dentro)

Exemplo

Testando vizinhança 13x13
(ordem de visita)



Testando vizinhança 13x13
(de fora para dentro)

Substituição em blocos

- Em vez de substituir os pixels um a um, podemos fazer isso em pequenos blocos (3x3, 5x5, etc.).
  - Melhor coerência local.
  - A fronteira entre os blocos pode ficar visível.
    - Normalmente são usados blocos pequenos.
  - A vizinhança testada precisa ser maior do que os blocos substituídos!

Exemplo

Bloco 3x3,
testando vizinhança 13x13

Bloco 5x5,
testando vizinhança 15x15

Exemplo

Bloco 9x9,
testando vizinhança 19x19

Bloco 13x13,
testando vizinhança 23x23

Ideia #3: comentários

- A abordagem com geração de textura tem alto custo computacional.
  - Alguns trabalhos usam estratégias para tentar melhorar esse aspecto:
    - Se um pixel tem um vizinho que já foi substituído por um valor vindo de uma posição (x,y), a sua substituição deve vir de uma posição próxima a (x,y).
    - Abordagem probabilística: em vez de testar todas as possibilidades, calcula o score apenas para um certo número de posições sorteadas.

Exemplos

- Vejamos alguns exemplos que comparam os resultados obtidos pelas 3 abordagens, e outras ideias!

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ