# SEAMLESS CLONING


NHÓM K17:

- **DIỆP HẢI BÌNH** **1712291**
- **NGUYỄN THANH BÌNH** **1712295**
- **HOÀNG GIA BẢO** **1712284**
- **NGUYỄN HOÀNG TUẤN CƯỜNG** **1712309**

# Input Images

source image

target image

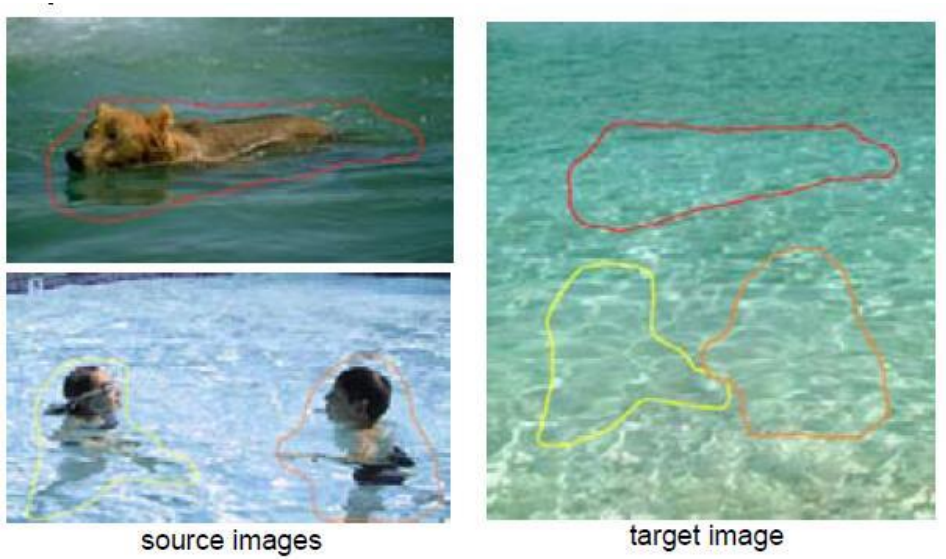
# Editing Results

Simple Cloning Result

Poisson Seamless Cloning

# Motivation

- Vấn đề đặt ra: làm thế nào để ghép các vật thể từ một ảnh này (ảnh nguồn) sang một ảnh khác (ảnh đích) mà vẫn giữ được độ chân thực?



source images

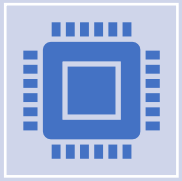
target image

simple cloning

Poisson seamless cloning

# Về mặt khoa học

Phương pháp cơ bản nhất để ghép vật thể từ ảnh nguồn vào ảnh đích là ***thay đổi các pixel của vị trí được chọn trên ảnh đích và thay thế bởi các pixel của vật thể trên ảnh nguồn.***

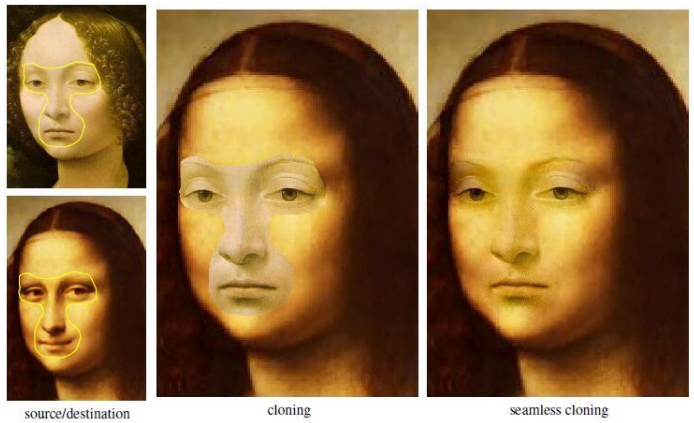


Phương pháp tốt hơn là **SEAMLESS CLONING**, ngoài thay thế các pixel ảnh đích bằng pixel của ảnh nguồn thì phương pháp này còn ***điều chỉnh các thông số*** của pixel sao cho phù hợp với bối cảnh xung quanh nhất.

Ngoài ra còn xóa đi các viền, cạnh bị thừa trong quá trình cắt vật thể.

# Về mặt ứng dụng

## Mô phỏng




## Chỉnh sửa ảnh

## Chuyển sang ảnh cartoon

## Thay đổi thuộc tính ảnh

# Ứng dụng khoa học

- Ứng dụng vào tiền xử lý dữ liệu
- Phân tích dữ liệu
- Khai thác dữ liệu ảnh/video
- Đa dạng nguồn dữ liệu phục vụ cho nghiên cứu và sáng tạo

# Ứng dụng thực tế

- Điện ảnh
- Mạng xã hội
- App mobile: ứng dụng ghép ảnh
- Giáo dục -> tăng tính sáng tạo cho trẻ

# 2/ Phát biểu bài toán

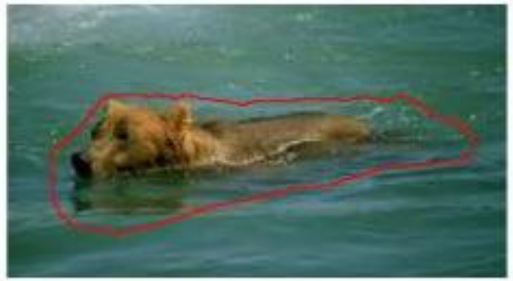
- Mục tiêu: muốn trộn đối tượng từ ảnh nguồn vào ảnh đích cách liền mạch, không gây ra “sự giả tạo”

**“Sự giả tạo” này là gì?**

- Con người nhạy cảm với sự thay đổi hơn là những thứ gì đó cố định
- Mắt chúng ta cũng thế, nó nhạy cảm với sự thay đổi của pixel so với các pixel xung quanh(gradient) hơn là giá trị màu của pixel đó

→ “sự giả tạo” ở đây chính là sự thay đổi quá lớn của gradient tại các pixel

**Vậy,** ý tưởng của Seamless Cloning là: thay vì sao chép giá trị của các pixel từ ảnh nguồn sang ảnh đích, ta sao chép gradient của nó

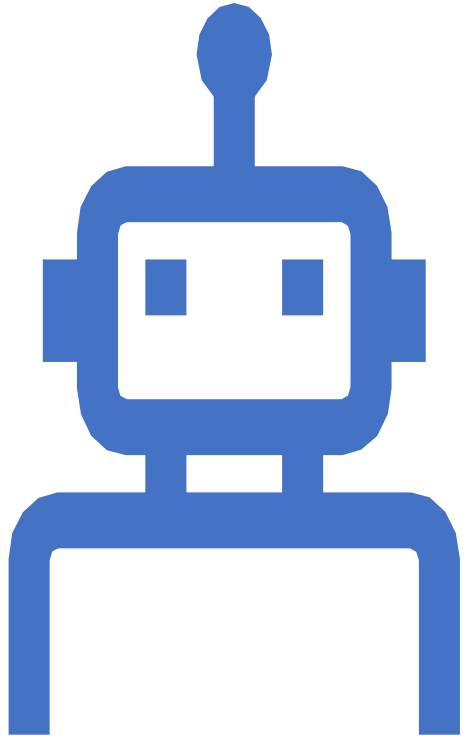


# 2/ Phát biểu bài toán

# 2/ Phát biểu bài toán

**Input:**

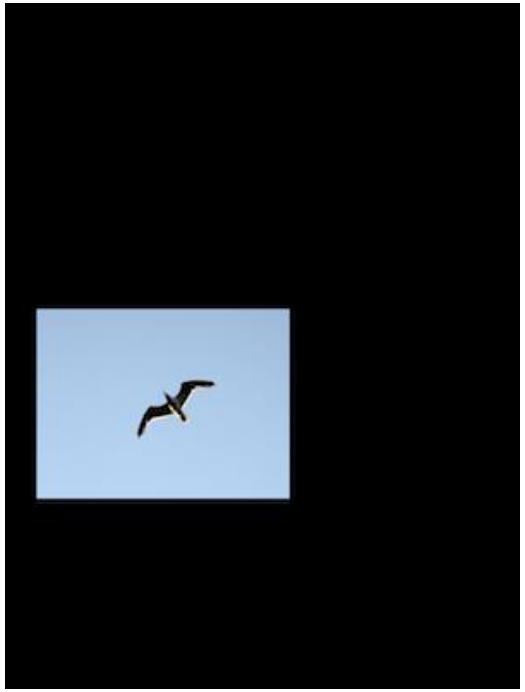
- Ảnh nguồn(S)
- Mask của ảnh nguồn(M) - để xác định đối tượng trong ảnh S muốn ghép vào ảnh T. Đây là ảnh nhị phân
- Ảnh đích/background(T)



**Output:**

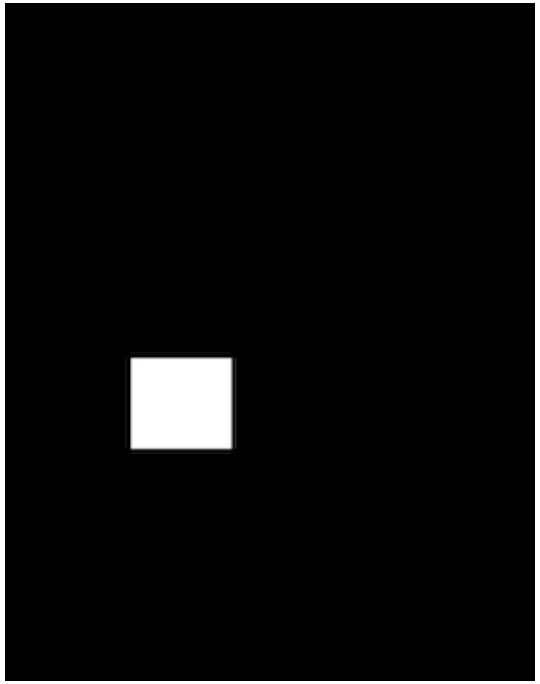
Ảnh kết quả(I) là ảnh sau khi đã trộn đối tượng trong ảnh S vào ảnh T. Ảnh I có kích thước bằng kích thước ảnh T

# 2/ Phát biểu bài toán



S

\+



M

\+

T

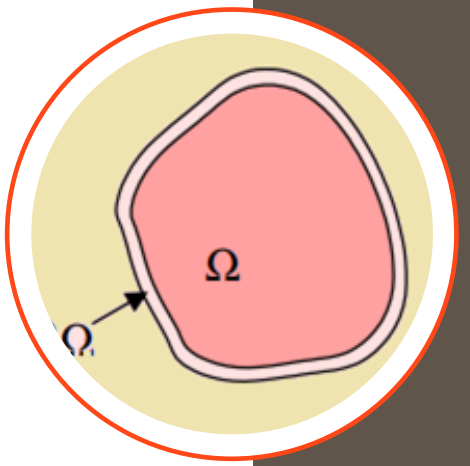
=

I

# 2/ Phát biểu bài toán

**Region(R) == Delta Omega + Omega**

**Phương pháp Poisson**




**Ý tưởng:**

- Ta muốn gradient của các pixel trong vùng R của ảnh kết quả I có giá trị gần nhất với gradient của các pixel tương ứng của ảnh nguồn(S), mà không thay đổi bất kì pixel nào của ảnh background.
- Hay nói cách ngắn gọn: ta cố gắng bảo toàn tối đa thông tin gradient của ảnh nguồn trên vùng R, mà không thay đổi bất kì pixel nào của ảnh background

# 2/ Phát biểu bài toán

## Phương pháp Poisson

Biểu diễn ý tưởng bằng toán học:

$$\min_{(x,y)\in R} \iint_R \|\nabla I(x,y) - \nabla S(x,y)\|^2 \, dxdy$$

Để làm việc này, dựa vào Euler-Lagrange, ta cần giải phương trình:

$$\nabla^2 I(x,y) = \nabla^2 S(x,y)$$

**Phương trình Poisson**

# 2/ Phát biểu bài toán

## Phương pháp Poisson

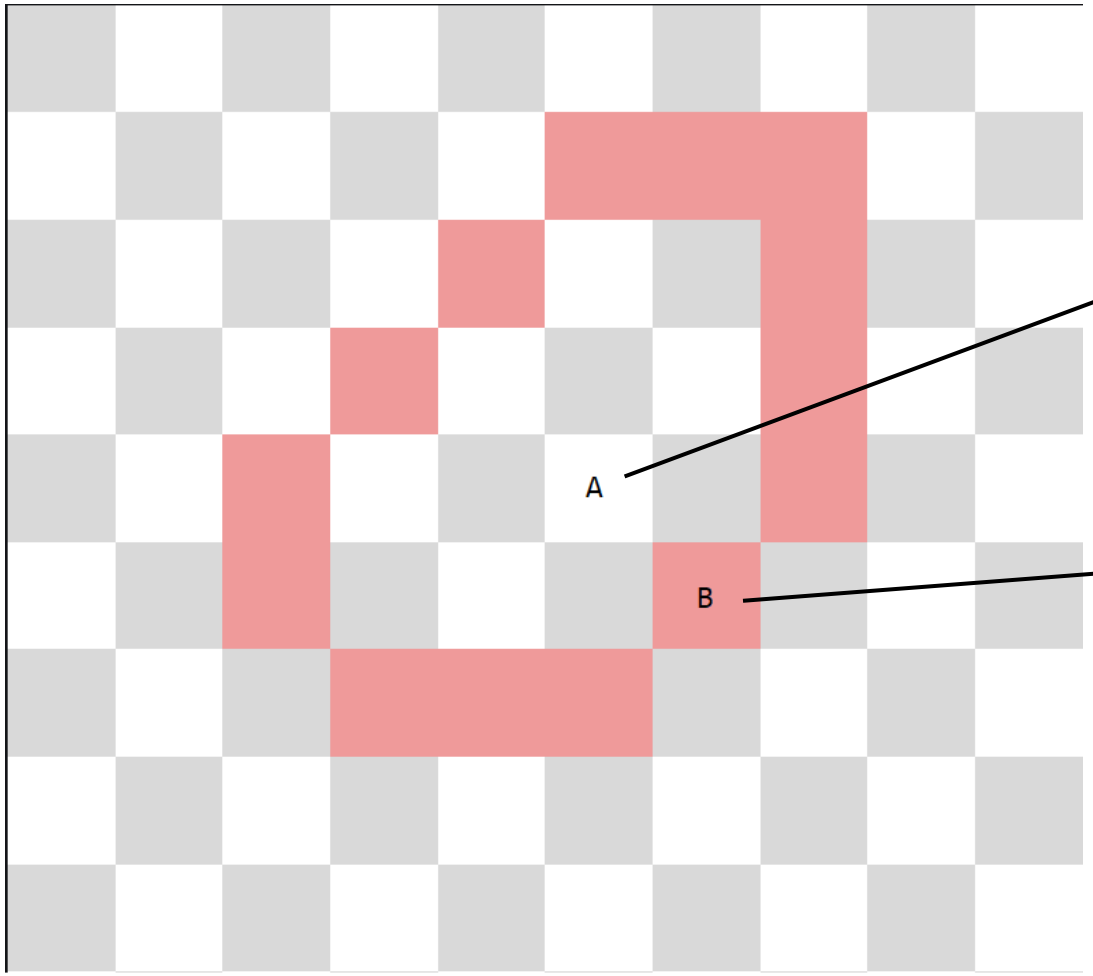
Giải pháp trên áp dụng trên hàm liên tục. Tuy nhiên, ảnh trong thực tế lại ở miền rời rạc. Vì thế giải pháp của ta phải thay đổi để phù hợp với ảnh rời rạc.

$$\nabla^2 I(x,y) = \nabla^2 S(x,y)$$

Toán tử Laplace, dùng filter $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

# 2/ Phát biểu bài toán

## Phương pháp Poisson




**R == Delta Omega + Omega**

**TH1**: điểm ảnh cần tìm nằm trên vùng Omega - nghĩa là 4 lân cận (trên, dưới, trái, phải) của nó thuộc vùng R

**TH2**: điểm ảnh cần tìm nằm trên vùng Delta Omega (những điểm tô hồng) - nghĩa là tồn tại ít nhất 1 lân cận (trên, dưới, trái, phải) nằm ngoài vùng R

# 2/ Phát biểu bài toán

## Phương pháp Poisson

**Giải phương trình:** $\nabla^2 I(x,y) = \nabla^2 S(x,y)$

**TH1:** Giả sử xét điểm ảnh A – nằm tại vị trí ($x_A$, $y_A$), tượng trưng cho các điểm ảnh thuộc TH1

$$\Leftrightarrow \quad I(x_A+1,\ y_A) + I(x_A-1,\ y_A) + I(x_A,\ y_A-1) + I(x_A,\ y_A+1) - 4.I(x_A,\ y_A) = S(x_A+1,\ y_A) + S(x_A-1,\ y_A) + S(x_A,\ y_A-1) + S(x_A,\ y_A+1) - 4.S(x_A,\ y_A)$$

**Vế trái là các ẩn**

**Vế phải đã biết**

# 2/ Phát biểu bài toán

## Phương pháp Poisson

**Giải phương trình:** $\nabla^2 I(x,y) = \nabla^2 S(x,y)$

**TH2:** Giả sử xét điểm ảnh B – nằm tại vị trí ($x_B$, $y_B$), tượng trưng cho các điểm ảnh thuộc TH2

$\Leftrightarrow$ $I(x_B+1,\ y_B) + I(x_B-1,\ y_B) + I(x_B,\ y_B-1) + I(x_B,\ y_B+1) - 4.I(x_B,\ y_B) = S(x_B+1,\ y_B) + S(x_B-1,\ y_B) + S(x_B,\ y_B-1) + S(x_B,\ y_B+1) - 4.S(x_B,\ y_B)$

Đã biết: $= T(x_B+1, y_B)$

Đã biết: $= T(x_B, y_B+1)$

---

$\Rightarrow$ $I(x_B-1, y_B) + I(x_B, y_B-1) - 4.I(x_B, y_B) = S(x_B+1,\ y_B) + S(x_B-1,\ y_B) + S(x_B,\ y_B-1) + S(x_B,\ y_B+1) - 4.S(x_B,\ y_B) - (T(x_B+1, y_B) + T(x_B, y_B+1))$

# 2/ Phát biểu bài toán

**Phương pháp Poisson**

**Nhận xét:**

Với N pixel thuộc vùng A, ta có được N phương trình để tính giá trị pixel I(x, y)

→Vậy bài toán trở về: **tính hệ N ẩn đã biết N phương trình**

→Nghĩa là giải phương trình ma trận tuyến tính: **A.X = B** để tìm ma trận X, khi đã biết ma trận A và B

**Trong đó:**

A: ma trận vuông NxN, chứa hệ số. Mỗi dòng gồm các số 0, -4, 1

X: ma trận Nx1, gồm các giá trị pixel I(x, y) cần tìm

B: ma trận Nx1, là giá trị vế phải trong 2 trường hợp bên trên

# 2/ Phát biểu bài toán

## Phương pháp Poisson

**Kết luận:**

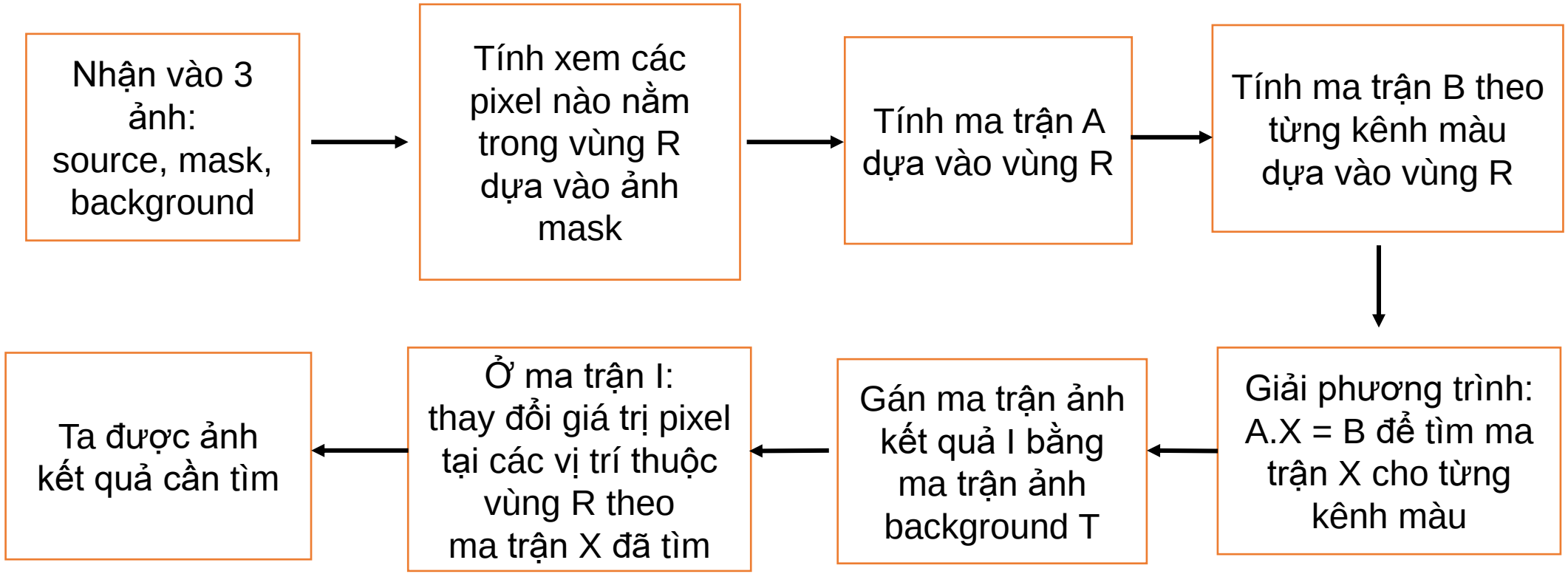
Vậy cuối cùng, để làm phương pháp Poisson, ta cần làm các việc sau:

- Cấu hình ma trận A và B theo 2 trường hợp trên
- Thực hiện giải phương trình ma trận tuyến tính: A.X = B để tìm ma trận X

→Ma trận X chính là giá trị các điểm ảnh cần tìm trong vùng A

# 2/ Phát biểu bài toán

## Framework




- Tốn tài nguyên lưu trữ, do ma trận có chiều: (NxN) – N là số pixel thuộc vùng R. Nếu vùng này 100x100 → Ma trận A có chiều: (10000 x 10000): khá lớn
- Cần thuật toán giải phương trình tuyến tính có độ phức tạp tốt để tăng tốc độ thực thi khi ma trận A đủ lớn

# Thách thức

# Related work

# Efficient Poisson Image Editing Using Pyramid

- Ý tưởng: Để cải thiện thời gian thực hiện tổng hợp hình ảnh sử dụng PIE, ta sử dụng cách chia để trị

# Efficient Poisson Image Editing Using Pyramid

Phương pháp: Ta thực hiện theo 3 cấp độ của kim tự tháp

Cấp độ 1: Ảnh nguồn và ảnh đích được kết hợp với nhau để tạo ra hình ảnh tổng hợp. Sử dụng PIE để tổng hợp

Cấp độ 2: Phân chia vùng ảnh cần ghép thành các mảnh nhỏ hơn của vùng ảnh, và tính toán trên từng mảnh nhỏ này

Cấp độ 3: Lặp lại cấp độ 2

# Efficient Poisson Image Editing Using Pyramid

Độ phức tạp của PIE là $O\left(\frac{2}{3}n^3\right)$ với n là kích thước của vùng ảnh cần ghép

Vì vùng ảnh được chia thành nhiều mảng nên ta có độ phức tạp của từng mảnh là $\sum_{i=1}^{r}\frac{2}{3}m_i^3 = \frac{2}{3} = \frac{2}{3}(m_1^3 + m_2^3 + \ \ldots + m_r^3) = T_1$

Thời gian tích toán nguyên vù$\sum_{i=1}^{r} m_i^3$ ng ảnh là $T = \frac{2}{3}n^3$

Vì $n = \ \sum_{i=1}^{r} m_i$ Nên *T* trở thành:

$$T = \frac{2}{3}(\sum_{i=1}^{r} m_i)^3 \ = \frac{2}{3}(m_1 + m_2 + \ \ldots + m_r)^3$$

# Efficient Poisson Image Editing Using Pyramid

Từ đó suy ra:

$$T = \frac{2}{3}\begin{pmatrix} m_1^3 + m_2^3 + \ \ldots + m_r^3) + (3m_1^2 m_2 + 3m_1^2 m_3 + \ \ldots + 3m_1^2 m_r + 3m_2^2 m_1 + 3m_2^2 m_3 \\ + \ \ldots + 3m_2^2 m_r + 3m_r^2 m_1 + 3m_r^2 m_2 + \ \ldots + 3m_r^2 m_{r-1} + 6m_1 m_2 \ldots m_r) \end{pmatrix}$$

$$= T_1 + T_2$$

Theo như biểu thức ở trên thì $T_1 < T$ một khoảng là $T_2$. Vì vậy chi phí thời gian để giải vùng ảnh sau khi chia thành các vùng nhỏ thấp hơn chi phí thời gian để giải nó tại một thời điểm.

# Bảng

| Tên phương pháp | Năm | Nguyên lý | Dữ liệu thử nghiệm | Độ đo |
|---|---|---|---|---|
| PRTS | 2015 | Chia vùng ẩn thành từng lát mỏng để xử lý | + ảnh bóng bay<br>+ ảnh bò sữa<br>+ ảnh chim<br>+ ảnh vịt | MSE<br>SSIM |
| PRSSB | 2015 | Chia vùng ẩn thành lưới các ô vuông nhỏ để xử lý | + ảnh bóng bay<br>+ ảnh bò sữa<br>+ ảnh chim<br>+ ảnh vịt | MSE<br>SSIM |

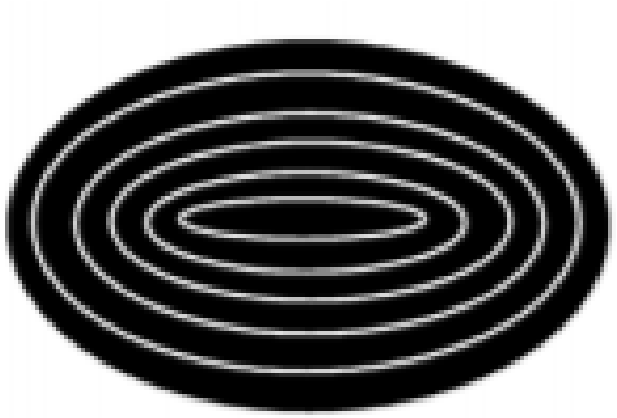
# Bảng thống kê số liệu theo hiệu suất và thời gian

| Method | Quality (%) |
|---|---|
| **PIE [14]** | 65 |
| **MVC [20]** | 67 |
| **SICDII [22]** | 70 |
| **PRTS2** | 71 |
| **PRTS3** | 70 |
| **PRSSB2** | 68 |
| **PRSSB3** | 67 |

| Method | Balloon | Bird | Cow | Phalarope |
|---|---|---|---|---|
| **PIE [14]** | 518.47 | 24.73 | 88.33 | 13.28 |
| **MVC[20]** | 183.49 | 15.61 | 42.38 | 9.44 |
| **SICDII[22]** | 100.96 | 9.1088 | 24.02 | 7.86 |
| **PRTS2** | **56.06** | **7.03** | **13.73** | **3.61** |
| **PRTS3** | **38.51** | **7.14** | **16.18** | **4.1** |
| **PRSSB2** | **36.60** | **4.25** | **11.24** | **1.88** |
| **PRSSB3** | **12.24** | **2.56** | **4.51** | **1.86** |

# Partitioning the unknown Region into Thin Slices (PRTS)

Phương pháp này chia vùng ảnh cần ghép đối tượng vào thành các lát cắt theo vòng từ ngoài vào trong. Sau đó, áp dụng phương trình Poisson để giải cho từng lát cắt riêng biệt lấy điều kiện biên Dirichlet từ hình ảnh tổng hợp là kết quả của cấp kim tự tháp trước đó



# PRTS

PRTS Algorithm: chia vùng không xác định thành các lát cắt mỏng

- Input: phần mask của vùng cần ghép, ảnh nguồn và đích

- Output: Ảnh đã được ghép chung lại với nhau

# PRTS

Thuật toán

1. Xây dựng các cấp kim tự tháp cho các hình ảnh nguồn, đích và mask bằng phương pháp kim tự tháp Gaussian
2. Tổng hợp ảnh nguồn và ảnh đích ở cấp độ thứ ba của kim tự tháp bằng phương pháp PIE
3. Ở cấp độ thứ hai của kim tự tháp ta làm như sau:
   - ❖ Cắt vùng ảnh cần ghép thành các lát mỏng theo các bước sau:
     - Dùng phương pháp erosion để xác định các lát cần cắt
     - Sử dụng phép XOR giữa vùng được xác định ở trên với cái vùng ảnh cần ghép gốc

# PRTS

Thuật toán

3. Ở cấp độ thứ hai của kim tự tháp ta làm như sau:

- ❖ Cắt vùng ảnh cần ghép thành các lát mỏng theo các bước sau:
  - Dùng phương pháp erosion để xác định các lát cần cắt
  - Sử dụng phép XOR giữa vùng được xác định ở trên với cái vùng ảnh cần ghép gốc
- ❖ (For) Với mỗi lát cắt ta thực hiện:
  - Tổng hợp ảnh nguồn và ảnh đích bằng phương pháp PIE
  - Giải phương trình Poisson với điều kiện biên Dirichlet lấy từ ảnh tổng hợp được tạo ở bước 1.

  End for

4. Tính hình ảnh tổng hợp cuối cùng bằng cách lặp lại bước trước đó ở cấp đầu tiên của kim tự tháp

# Partitioning the unknown Region into Small Square Blocks (PRSSB)

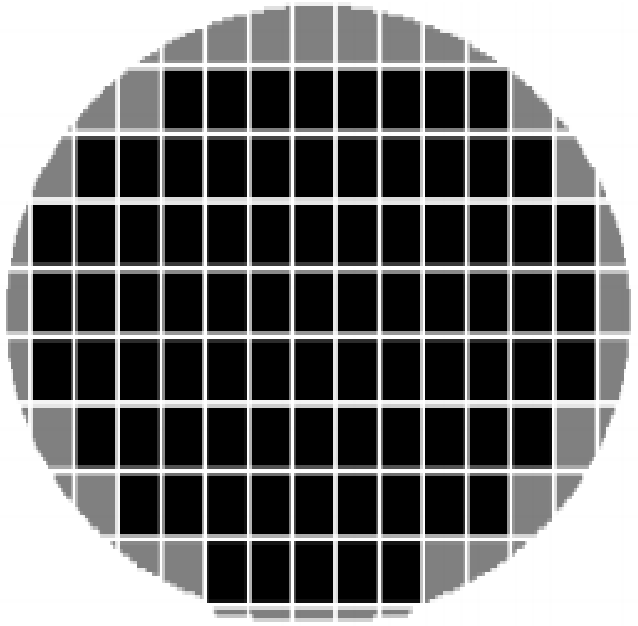


**Figure 3.** The unknown region is divided into small square blocks

# PRSSB

- Input: phần mask của vùng cần ghép, ảnh nguồn và đích
- Output: Ảnh đã được ghép chung lại với nhau

# PRSSB – Thuật toán

1. Xây dựng các cấp kim tự tháp cho các hình ảnh nguồn, đích và mask bằng phương pháp kim tự tháp Gaussian
2. Tổng hợp ảnh nguồn và ảnh đích ở cấp độ thứ nhất của kim tự tháp bằng phương pháp PIE
3. Ở cấp độ thứ hai của kim tự tháp ta làm như sau:
   a. Cắt vùng ảnh cần ghép thành lưới các ô vuông có kích thước BS.
   b. Tạo ma trận SA theo các bước sau:
      i. Kích thước ma trận là $BS^2 \text{x}\ BS^2\ trong$ đó: $BS^2$: $số\ pixel\ trong\ lưới$

# PRSSB – Thuật toán

For r = (0, BS-1):

For c = (0,BS-1):

Xác định bộ số lân cận của pixel P(r,c)

Lặp qua mỗi pixel lân cận:

if nằm trên viền Dirichlet

SA(r,c)=0

else

SA(r,c)= -1

# PRSSB – Thuật toán

- Với mỗi 1 ô vuông trong lưới ô trên:
  - Thực hiện phương pháp PIE
  - Giải phương trình poisson với điều kiện Dirichlet
  - Nếu tất cả các pixel là ẩn
    - Giải ma trận SA
  - Nếu không thì giải ma trận A như bình thường

➢Lặp lại các bước trên 1 lần nữa tại mức 3 của “pyramid”