

ĐẠI HỌC QUỐC GIA TP HCM  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN  
KHOA CÔNG NGHỆ THÔNG TIN

# XỬ LÝ ẢNH SỐ & VIDEO SỐ

## BÁO CÁO ĐỒ ÁN

**Đề tài:** Seamless Cloning

**Giảng viên hướng dẫn:** PGS.TS Lý Quốc Ngọc

**Thực hiện:** Nhóm K17

Diệp Hải Bình	1712291
Hoàng Gia Bảo	1712284
Nguyễn Thanh Bình	1712295
Nguyễn Hoàng Tuấn Cường	1712309

**-TPHCM 01/2021-**

# MỤC LỤC

<b>I. Giới thiệu seamless cloning</b>	3
1. Động lực thực hiện	3
2. Các vấn đề của bài toán	4
Mục tiêu	4
Input, output của bài toán	5
Phương pháp Poisson Image Editing	5
Kết luận	8
Framework	8
Thách thức của bài toán	9
<b>II. Các nghiên cứu liên quan</b>	9
Chỉnh sửa ảnh hiệu quả bằng Pyramid	9
1. Partitioning the unknown Region into Thin Slices (PRTS)	10
2. Partitioning the unknown Region into Small Square Blocks (PRSSB)	11
Kết quả tổng hợp	13
<b>III. Nguồn tham khảo</b>	16

# I. Giới thiệu seamless cloning

## 1. Động lực thực hiện

### Về khoa học

- ✚ Qua các thao tác cơ bản nhất là thao tác trên các pixel của ảnh, ta có rất nhiều thuật toán trong xử lý ảnh như: tăng giảm độ sáng, phát hiện biên cạnh, khử nhiễu,....

### Về ứng dụng

- ✚ Giảm chi phí, thời gian trong quá trình làm phim

Trong sản xuất phim, thay vì quay thực tế tốn chi phí, thời gian nhiều thì có thể thông qua thuật toán này ghép các vật thể hay thậm chí con người vào các bối cảnh khác nhau mà độ chân thực vẫn được đảm bảo.

- ✚ Thực hiện các cảnh giả tưởng, không có trong thực tế

Trong các bộ phim về khoa học viễn tưởng, một số cảnh đòi hỏi phải có sự can thiệp của sự chỉnh sửa chứ không thể quay theo cách truyền thống.

- ✚ Làm hoàn hảo một tấm ảnh

Cụ thể là một số tấm ảnh có một số chi tiết không hoàn hảo có thể là do lỗi người chụp hoặc lỗi thiết bị, qua thuật toán này có thể bổ sung, chỉnh sửa lại sao cho đẹp nhất, hoàn hảo nhất.

- ✚ Ngoài ra còn các ứng dụng mang tính thương mại như

Hoàn đổi khuôn mặt (Face Swapping): ứng dụng đã rất hot vào khoảng 2 đến 3 năm trước

Các tính năng ghép mặt tích hợp trên ứng dụng Messenger, Tiktok,.....

Một số hình ảnh có sử dụng kỹ thuật Poisson Image Editing



## 2. Các vấn đề của bài toán



### Mục tiêu của bài toán

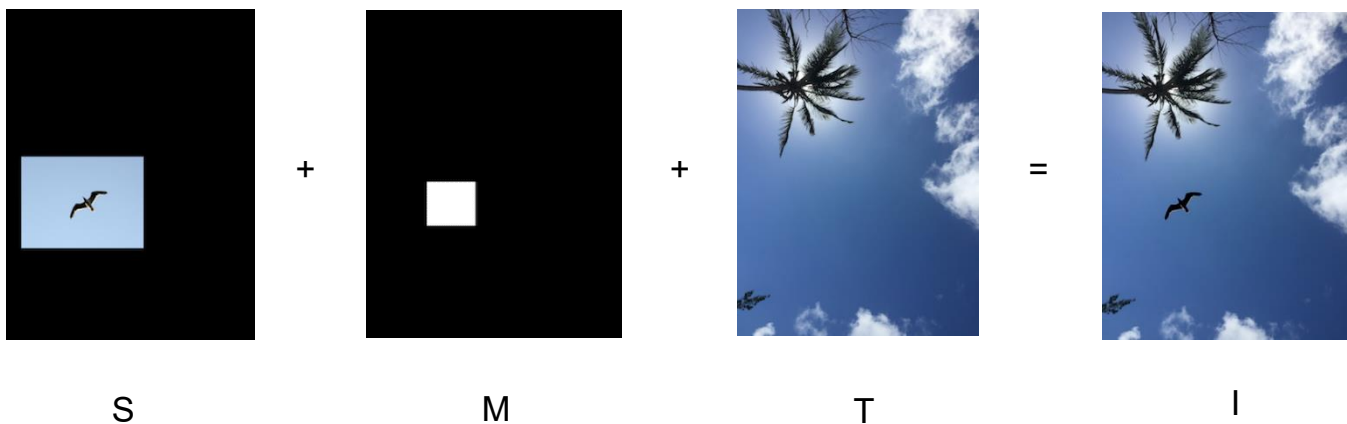
- ✚ Muốn trộn đối tượng từ ảnh nguồn vào ảnh đích một cách liền mạch, không gây ra “sự giả tạo”
- ✚ “Sự giả tạo” này là gì?
  - Con người nhạy cảm với sự thay đổi hơn là những thứ gì đó cố định.
  - Mắt chúng ta cũng thế, nó nhạy cảm với sự thay đổi của pixel so với các pixel xung quanh (gradient) hơn 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.

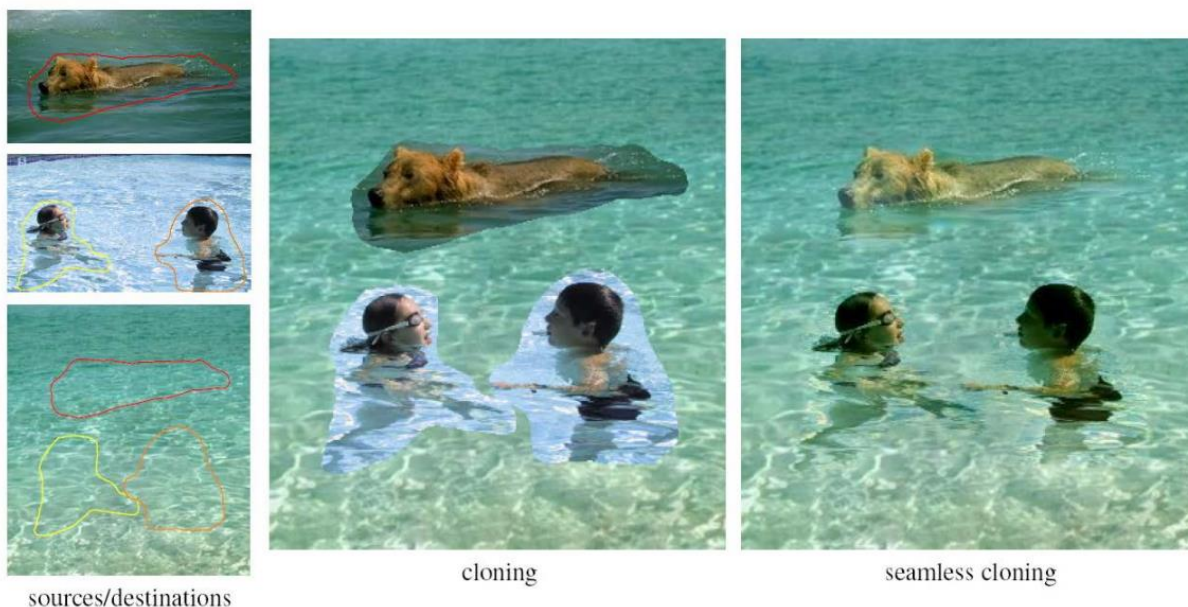
## Input, output của bài toán

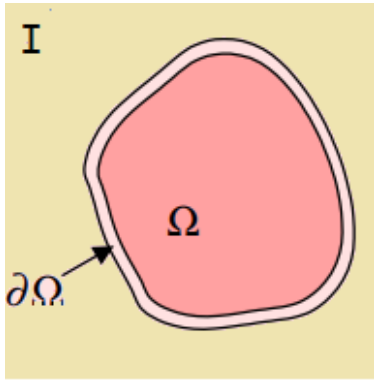
-  **Input: 3 ảnh cùng kích thước**
- Ảnh nguồn (S)
  - Mask của ảnh nguồn(M) - để xác định đối tượng trong ảnh S muốn ghép vào ảnh T. Đây là ảnh nhị phân.
  - Ảnh đích/background (T)
-  **Output:** Ảnh kết quả (I) là ảnh sau khi đã trộn đối tượng trong ảnh S vào ảnh T. Ảnh I có kích thước bằng kích thước ảnh T.



Ảnh mask (M) có vùng cần trộn (vùng màu trắng) tương ứng vừa là vị trí đối tượng trong ảnh S muốn trộn vào ảnh T, vừa là nơi đặt đối tượng đó trong ảnh T (để dễ cài đặt thuật toán).

## Phương pháp Poisson Image Editing





$$R = \partial\Omega + \Omega$$

### Ý tưởng:

- Ta muốn gradient của các pixel trong vùng **R** (chính là vùng màu trắng trong ảnh M) 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 (T).
- 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 (T)

Biểu diễn 2 ý tưởng trên bằng toán học

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

Để giải bài toán này, dựa vào Euler-Lagrange, đồng nghĩa với việc ta giải phương trình:

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

→ Đây là phương trình Poisson nên phương pháp có tên là Poisson

Giải pháp đưa ra bên trên áp dụng vào miền 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.

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

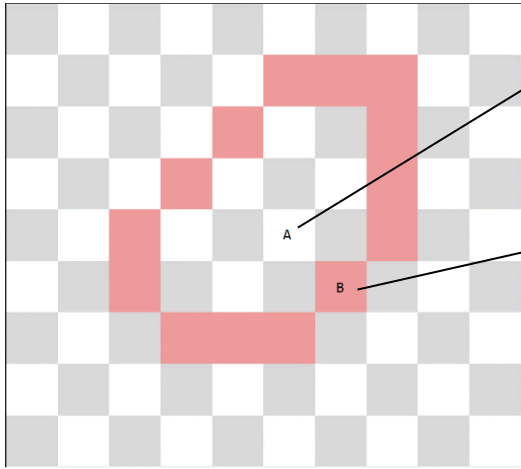
→ Đây chính là Laplacian operator, ta có thể tính xấp xỉ nó trong miền rời rạc bằng Laplace filter:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

**Xảy ra 2 trường hợp sau:**

$$R = \partial\Omega + \Omega$$

- $\partial\Omega$ : Những điểm tô hồng (điểm biên)
- $\Omega$ : Những điểm nằm bên trong



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

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

Giải phương trình:

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

**Trường hợp 1:** Giả sử xét điểm ảnh A – nằm tại vị trí  $(x_A, y_A)$ , tượng trưng cho các điểm ảnh thuộc trường hợp 1 (TH1)

$$\begin{aligned} I(x_A + 1, y_A) + I(x_A - 1, y_A) &= S(x_A + 1, y_A) + S(x_A - 1, y_A) \\ + I(x_A, y_A - 1) + I(x_A, y_A + 1) &= + S(x_A, y_A - 1) + S(x_A, y_A + 1) \\ - 4 \cdot I(x_A, y_A) &= - 4 \cdot S(x_A, y_A) \end{aligned}$$

Vế trái là các ẩn

Vế phải có thể tính được

**Trường hợp 2:** Giả sử xét điểm ảnh B – nằm tại vị trí  $(x_B, y_B)$ , tượng trưng cho các điểm ảnh thuộc TH2 (cách xử lý các pixel nằm ở biên  $\partial\Omega$ )

$$\begin{aligned} I(x_B + 1, y_B) + I(x_B - 1, y_B) &= S(x_B + 1, y_B) + S(x_B - 1, y_B) \\ + I(x_B, y_B - 1) + I(x_B, y_B + 1) &= + S(x_B, y_B - 1) + S(x_B, y_B + 1) \\ - 4 \cdot I(x_B, y_B) &= - 4 \cdot S(x_B, y_B) \\ \hline = T(x_B + 1, y_B) &= T(x_B, y_B + 1) \end{aligned}$$



$$\begin{aligned}
 I(x_B - 1, y_B) &+ I(x_B, y_B - 1) - 4 \cdot I(x_B, y_B) \\
 &= S(x_B + 1, y_B) + S(x_B - 1, y_B) + S(x_B, y_B - 1) + S(x_B, y_B + 1) - (T(x_B + 1, y_B) + T(x_B, y_B + 1)) - 4 \cdot S(x_B, y_B)
 \end{aligned}$$

### Nhận xét:

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

→ Vậy bài toán trở về dạng **tính hệ  $N$  ẩn đã biết  $N$  phương trình**. Nghĩa là giải phương trình ma trận tuyến tính  $A \cdot X = B$  để tìm ma trận  $X$ , khi đã biết ma trận  $A$  và  $B$ .

### **Trong đó:**

$A$ : ma trận ( $N \times N$ ), với  $N$  là số pixel của vùng  $R$ . Mỗi dòng gồm số 0, -4, 1 (dựa theo Laplace filter)

$X$ : ma trận các ẩn số ( $N \times 1$ ) gồm có  $N$  pixel  $I(x, y)$  trong vùng  $R$  cần tìm

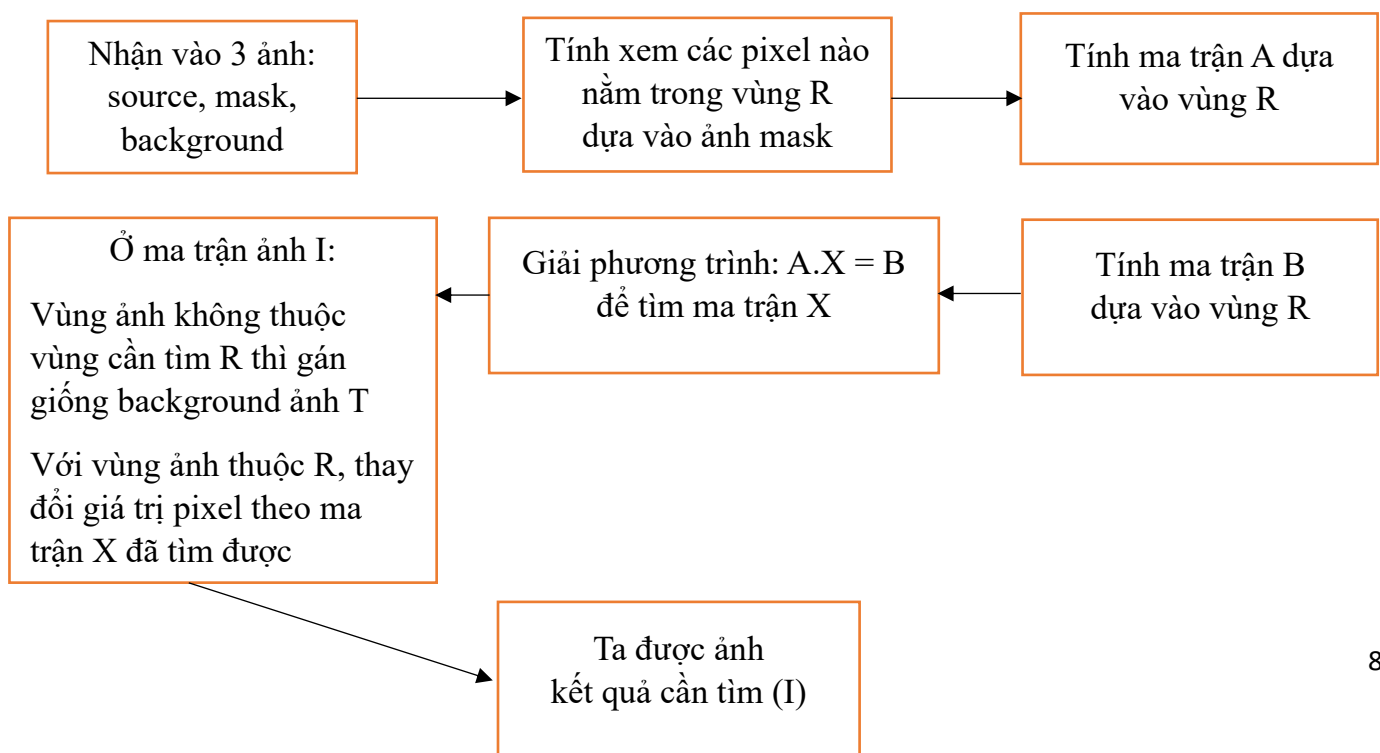
$B$ : là ma trận ( $N \times 1$ ), chứa giá trị về phải trong 2 trường hợp bên trên

### **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:  $A \cdot X = B$  để tìm ma trận  $X$ .
- Ma trận  $X$  chính là giá trị các điểm ảnh cần tìm trong vùng  $R$ .

### **Framework**





## Thách thức của bài toán

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

## II. Các nghiên cứu liên quan

### Chỉnh sửa ảnh Poisson hiệu quả bằng Pyramid

- ✚ Việc tạo thành hai ảnh dựa trên phương trình Poisson yêu cầu giải một hệ tuyến tính với  $n$  ẩn số để tính giá trị của  $n$  pixel. Trong công trình này, một phương pháp thành phần ảnh được cải tiến dựa trên phương trình Poisson được giới thiệu để giảm thời gian thực hiện trong quá trình ghép ảnh. Các phương pháp đề xuất giải phương trình Poisson bằng cách sử dụng kim tự tháp hình ảnh và các phương pháp chia đệ trị. Kim tự tháp hình ảnh và các phương pháp chia đệ trị được sử dụng để tăng tốc thực hiện nhiều loại hình ảnh khác nhau.
- ✚ Sử dụng các phương pháp đã đề xuất, thời gian để giải quyết một vấn đề lớn tại cùng một thời điểm thì lớn hơn so với thời gian thực hiện nếu nó được chia thành các vấn đề nhỏ hơn. Trong các phương pháp đề xuất, hai và ba cấp tháp được xây dựng để tăng hiệu quả của quá trình. Các phương pháp đề xuất bắt đầu từ cấp độ kim tự tháp thứ ba. Ở cấp thứ ba của kim tự tháp, hình ảnh nguồn và hình ảnh đích được kết hợp với nhau để tạo ra hình ảnh tổng hợp. Phương pháp PIE được sử dụng để tạo các ảnh nguồn và ảnh đích trong cấp độ thứ ba của kim tự tháp. Phương pháp này lấy các giá trị cường độ của các pixel ở viền thành các giá trị của hình ảnh đích. Sau đó, phương trình Poisson được giải để tính toán các giá trị cường độ cho các pixel trong vùng bên trong. Do đó, đường viền của đối tượng trong ảnh tổng hợp rất mịn. Sau khi tạo ra hình ảnh tổng hợp từ cấp kim tự tháp thứ ba, nó được sử dụng trong cấp thứ hai của kim tự tháp. Ở cấp độ thứ hai, ảnh nguồn và ảnh đích được kết hợp bằng cách giải phương trình Poisson lấy điều kiện biên Dirichlet từ kết quả ảnh tổng hợp từ cấp độ thứ ba. Vì vậy, hình ảnh tổng hợp được tạo ra từ cấp độ thứ hai là chân thực và mượt mà hơn nhiều. Ở cấp độ thứ hai, vùng chưa biết được chia thành một số vùng nhỏ chưa biết và hệ thống tuyến tính thừa thớt được giải cho từng vùng nhỏ riêng biệt. Cuối cùng, quá trình xảy ra ở cấp độ thứ hai được lặp lại ở cấp độ đầu tiên để tạo ra hình ảnh tổng hợp cuối cùng.
- ✚ Như đã giải thích trước đây, các phương pháp được đề xuất dựa trên việc phân chia vùng chưa biết thành một số vùng nhỏ giải quyết hệ thống tuyến tính thừa thớt cho từng vùng nhỏ riêng biệt. Đối với vùng chưa biết, độ phức tạp của phương pháp thừa số hóa Lower-Upper (LU) để giải hệ tuyến tính trong phương trình  $Ax = b$  là xấp xỉ  $O\left(\frac{2}{3}n^3\right)$  trong đó kích thước của  $A$  là  $n \times n$ . Nếu vùng chưa biết được chia thành các vùng nhỏ, thì chi phí thời gian để giải hệ trong phương trình  $Ax = b$  cho vùng chưa biết bằng cách sử dụng thừa số hóa LU bằng tổng

chi phí thời gian để giải nó cho từng vùng nhỏ riêng biệt, như được minh họa trong phương trình sau:

$$\sum_{i=1}^r \frac{2}{3} m_i^3 = \frac{2}{3} (m_1^3 + m_2^3 + \dots + m_r^3) = T_1$$

trong đó  $m_i$  là số pixel chưa biết trong vùng  $i$  và  $r$  là số vùng nhỏ. Độ phức tạp  $T$  để giải hệ thống tuyến tính cho tất cả các vùng một lần được cho bởi phương trình sau:

$$T = \frac{2}{3} n^3$$

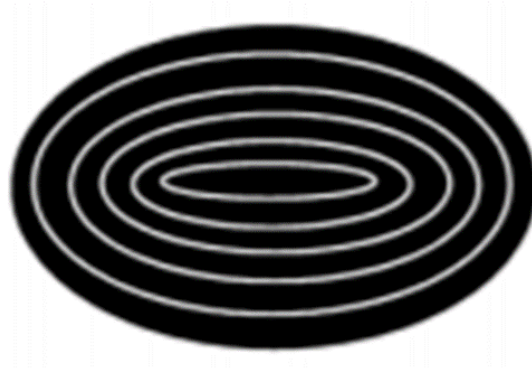
Từ đó suy ra:

$$T = \frac{2}{3} \left( m_1^3 + m_2^3 + \dots + m_r^3 + (3m_1^2 m_2 + 3m_1^2 m_3 + \dots + 3m_1^2 m_r + 3m_2^2 m_1 + 3m_2^2 m_3 + \dots + 3m_2^2 m_r + 3m_3^2 m_1 + 3m_3^2 m_2 + \dots + 3m_3^2 m_r + \dots + 3m_{r-1}^2 m_r + 6m_1 m_2 \dots m_r) \right) = T_1 + T_2$$

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.

## 1. 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 Algorithm: Hiệu quả của thuật toán này đạt được bằng cách chia nhỏ vùng ảnh thành những lát cắt mỏng từ ngoài vào trong.*

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

Output: Ảnh đã được ghép chung lại với nhau

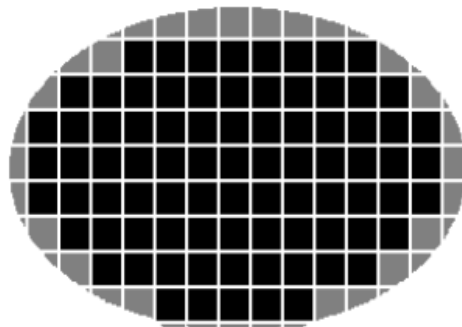
## Thuật toán

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
    - Tưởng tượng ảnh là một ma trận, thuật toán này lấy từng vòng pixel từ ngoài vào trong của ma trận, nên nó như cắt lát cắt từ ngoài vào trong
  - ✚ (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

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

Trong phương pháp này, vùng chưa biết được chia thành các khối vuông nhỏ. Hình dạng của các vùng trong ranh giới của vùng không xác định là hình vuông không hoàn toàn (không phải tất cả các pixel trong khối đều là ẩn số) và hình dạng của các vùng bên trong là hình vuông (tất cả các pixel là ẩn số), như trong hình sau:



The unknown region is divided into small square blocks

Sau đó, các phương trình Poisson được giải cho từng vùng riêng biệt, lấy điều kiện biên Dirichlet từ hình ảnh đầu ra của mức kim tự tháp trước đó. Các bước được mô tả trong thuật toán sau:

Thuật toán PRSSB: PIE hiệu quả dựa trên việc phân vùng không xác định thành các khối vuông nhỏ.

Inputs: mặt nạ (mask) của vùng chưa biết, hình ảnh nguồn và đích.

Output: hình ảnh tổng hợp cuối cùng.

1. Xây dựng các mức kim tự tháp cho các hình ảnh nguồn, đích và mặt nạ bằng cách sử dụng phương pháp kim tự tháp Gaussian.
  2. Soạn ảnh nguồn và ảnh đích ở mức thứ ba của hình chóp bằng phương pháp PIE.
  3. Ở cấp hai của hình chóp ta làm như sau:
    - a. Chia mặt nạ của vùng chưa biết thành các khối vuông nhỏ, mỗi khối có kích thước  $BS$ .
    - b. Tạo ma trận thừa thớt  $SA$  sẽ được sử dụng nếu tất cả các pixel trong khối là ẩn số bằng cách thực hiện các bước sau:
      - i. Khởi tạo ma trận  $SA$  với các phần tử size  $(BS^2 \times BS^2)$  và đường chéo 4s, trong đó  $BS^2$  là số pixel trong khối.
      - ii. for  $c = 0$  to  $BS-1$  do
 

for  $c = 0$  to  $BS-1$  do  
 Xác định tập hợp NB các pixel lân cận xung quanh pixel  $P(r,c)$   
 for  $px$  in NB1 do  
   if NB1 là pixel trên biên  
      $SA(r,c) = 0$   
   else  
      $SA(r,c) = -1$
      - c. For block in mask do
        - i. Soạn ảnh nguồn và ảnh đích bằng phương pháp PIE
        - ii. Giải phương trình Poisson với điều kiện biên Dirichlet lấy từ ảnh tổng hợp được tạo từ bước 1.
        - iii. if tất cả các điểm ảnh trên khối là ẩn số
 

Sử dụng ma trận  $SA$  được tính từ trước để giải hệ thống tuyến tính.
    - else
 

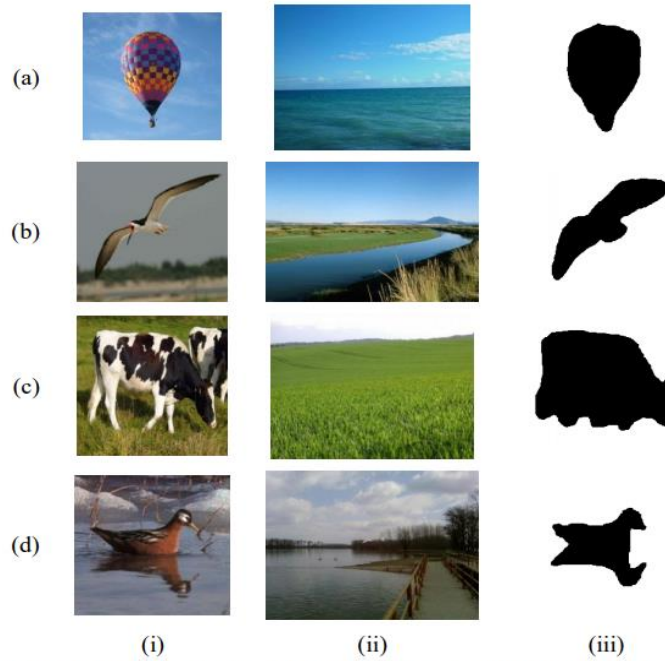
Tính ma trận thừa  $A$  để giải hệ thống tuyến tính.
4. Tính toán hình ảnh tổng hợp cuối cùng bằng cách lặp lại bước trước đó ở cấp đầu tiên của hình chóp.

## Kết quả tổng hợp

Bộ dữ liệu thực nghiệm:

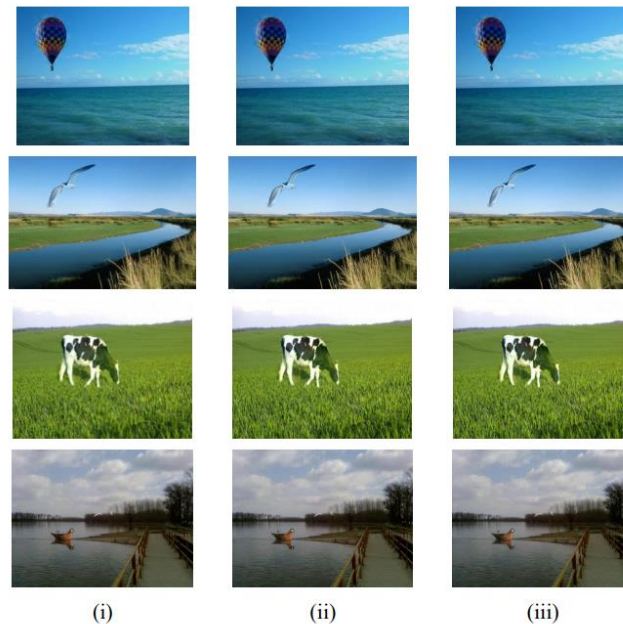
**Table 1.** Number of pixels in the unknown region for all instances

Instance	(a) Balloon	(b) Bird	(c) Cow	(d) Phalarope
Number of pixels	356,409	74,529	146,689	42,983



**Figure 4.** (i) Source image. (ii) Destination image. (iii) Object mask.

Kết quả thử nghiệm bộ dữ liệu trên bằng phương pháp truyền thống và 2 phương pháp cải tiến:



**Figure 5.** (i) Image produced by the PIE method [14]. (ii) Image produced by the PRSSB method. (iii) Image produced by the PRTS method.

**Bảng thống kê kết quả thực nghiệm trên nhiều phương pháp khác nhau:**

**A quality comparison between different methods**

<b>Method</b>	<b>Quality (%)</b>
<b>PIE [14]</b>	65
<b>MVC [20]</b>	67
<b>SICDII [22]</b>	70
<b>PRTS2</b>	71
<b>PRTS3</b>	70
<b>PRSSB2</b>	68
<b>PRSSB3</b>	67

- ✚ Chất lượng giữa các phương pháp không khác nhau nhiều tức sự cải tiến này không cải tiến được quá nhiều về mặt hiệu suất:
- ✚ PRSSB và PRTS chỉ hơn PIE truyền thống 2-3%. Tuy nhỏ nhưng cũng cho thấy phần nào sự tiến bộ trong thuật toán nhằm tăng độ chính xác.
- ✚ Thời gian mới là yếu tố chính được chú trọng của 2 phương pháp này:

**Table 12. A run time comparison between different methods**

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

- ✚ Ta dễ dàng nhận ra rằng PRSSB3 nhỏ hơn PIE ở tập Balloon tận 43 lần. Hoặc số lần giảm ít nhất là 3.4 lần. Sự rút ngắn thời gian vượt ngoài mong đợi.
- ✚ Qua đó cho ta thấy PRTS và PRSSB không làm tăng quá nhiều về mặt hiệu suất nhưng nó mang tính đột phá về thời gian chạy của thuật toán.

### **III. Nguồn tham khảo**

Poisson Image Editing, Patrick Pérez, Michel Gangnet, Andrew Blake

Efficient Poisson Image Editing, Khaled F.Hussain, Rasha M.Kamel, 2015