Bài thực hành: Giấu tin trong ảnh bằng thuật toán DCT

1. Mục đích

Bài thực hành này hướng dẫn sử dụng thuật toán DCT để giấu tin trong ảnh.

Thuật toán DCT (Discrete Cosine Transform – Biến đổi Cosin Rời rạc) là một phép biến đổi toán học dùng phổ biến trong xử lý ảnh, DCT chuyển tín hiệu từ miền không gian (spatial domain) sang miền tần số (frequency domain).

2. Yêu cầu đối với sinh viên

Có kiến thức cơ bản về giấu tin trong ảnh bằng thuật toán DCT.

Có kiến thức cơ bản về ngôn ngữ Python.

3. Nội dung thực hành

3.1 Khởi động bài lab

Cài đặt bài lab tại: *https://github.com/NgocTaif/stego_dct_code*

Giản nén và chuyển vào thư mục */labtainer/trunk/labs.*

Trên terminal, gõ:

labtainer -r stego_dct_code

Sau khi khởi động xong, một terminal ảo sẽ xuất hiện.

3.2 Tiền xử lý

Trên terminal sử dụng lệnh: *fim input.png* để xem ảnh được chuẩn bị tiến hành thực hiện giấu tin.

Trên terminal, tiếp tục gõ lệnh:

python3 preprocessor/create_gray_img.py input.png

Chương trình *create_gray_imgx.py* trong thư mục *preprocessor* chuyển đổi từ ảnh gốc ban đầu (*input.png*) thành ảnh xám (*img_gray.png*).

Ảnh xám này sẽ được thực hiện chuyển sang dạng ma trận điểm ảnh và tiến hành giấu tin.

Trên terminal, tiếp tục gõ lệnh:

python3 preprocessor/create_img_matrix.py img_gray.png

Chương trình *create_gray_imgx.py* trong thư mục *preprocessor* thực hiện chuyển đổi ảnh xám (*img_gray.png*) thành ma trận điểm ảnh (*image_matrix.txt*).

Thực hiện sử dụng *nano* để kiểm tra file ma trận điểm ảnh *image_matrix.txt*, nhận thấy các giá trị đều nằm trong đoạn [0, 255], các giá trị này đại diện cho độ sáng của điểm ảnh tại vị trí đó.

Qua bước tiền xử lý ta biết được kích thước của ảnh là 512 x 512 pixels. Trong quá trình giấu tin, ta sẽ chia ma trận điểm ảnh (*image_matrix.txt*) này thành các khối 8x8 pixels, có thể coi ma trận điểm ảnh (*image_matrix.txt*) là 1 ma trận điểm ảnh gồm 64 khối 8x8 pixels.

Ta sẽ thực hiện giấu các bit tin vào trong các khối ma trận 8x8 này.

3.3 Giấu tin

Trước tiên, ta cần thực hiện trích xuất các ma trận điểm ảnh có kích thước 8x8 từ ma trận điểm ảnh tổng trong *image_matrix.txt* để tiến hành giấu tin.

Trên terminal, ta sử dụng chương trình *extract_matrix.py* để trích xuất các ma trận khối 8x8 đó, gõ lệnh:

python3 extract_matrix.py image_matrix.txt <i> <j>

Trong đó *<i>, <j>* là các tham số đầu vào, chương trình *extract_matrix.py* sẽ lấy khối 8x8 tại vị trí hàng i cột j ở ma trận tổng khối 512 x 512 *image_matrix.txt* mà ta đã đề cập ở bước trên. Các ma trận khối 8x8 này khi trích xuất sẽ được lưu tại các file *matrix_8x8_vitri_<i>_<j>.txt* trong thư mục *extracted_matrix.*

Trong bài thực hành này, ta sẽ tiến hành trích xuất và giấu tin trong 6 khối ma trận 8x8 tại các vị trí (i, j) sau của ma trận tổng *image_matrix.txt*: *(0,0), (0,1), (0,2), (1,0), (1,1), (1,2).*

Tiếp tục quá trình giấu tin, ta sẽ sử dụng phép biến đổi DCT để chuyển đổi các hệ số giá trị từ miền không gian sang dạng tần số:

$$F(u,v) = \frac{C(u)C(v)}{4}\sum_{j=0}^{7}\sum_{k=0}^{7} f(j,k)\cos\frac{(2j+1)u\pi}{16}\cos\frac{(2k+1)v\pi}{16}$$

Trong đó: f(j ,k) là các mẫu của ảnh gốc trong khối 8x8 điểm ảnh.

F(u, v) là các hệ số của khối DCT 8x8.

Trên terminal, ta gõ các lệnh:

python3 dct_transform/dct_discrete_cosine.py extracted_matrix/matrix_8x8_vitri_<i>_<j>.txt

Chương trình *dct_discrete_cosine.py* trong thư mục *dct_transform* sẽ thực hiện biến đổi DCT lên từng ma trận khối 8x8 để chuyển đổi sang miền hệ số DCT. Biến đổi DCT sẽ tạo ra các hệ số DCT tương ứng với các tần số khác nhau trong ảnh.

Sau khi đã biến đổi DCT thành công các khối ma trận 8x8, ta tiếp tục thực hiện lượng tử hóa các khối ma trận này. Mục đích của việc lượng tử hóa là giảm số lượng bit cần để lưu trữ các hệ số biến đổi bằng việc giảm độ chính xác của các hệ số này cho nên lượng tử là quá trình xử lý có mất thông tin.

Trên terminal, tiếp tục thực hiện gõ các lệnh:

python3 dct_transform/dct_quantized.py
extracted_matrix/matrix_8x8_vitri_<i>_<j>.txt

Chương trình *dct_quantized.py* trong thư mục *dct_transform* sẽ tiến hành lượng tử hóa các ma trận khối 8x8 sau khi đã được biến đổi DCT.

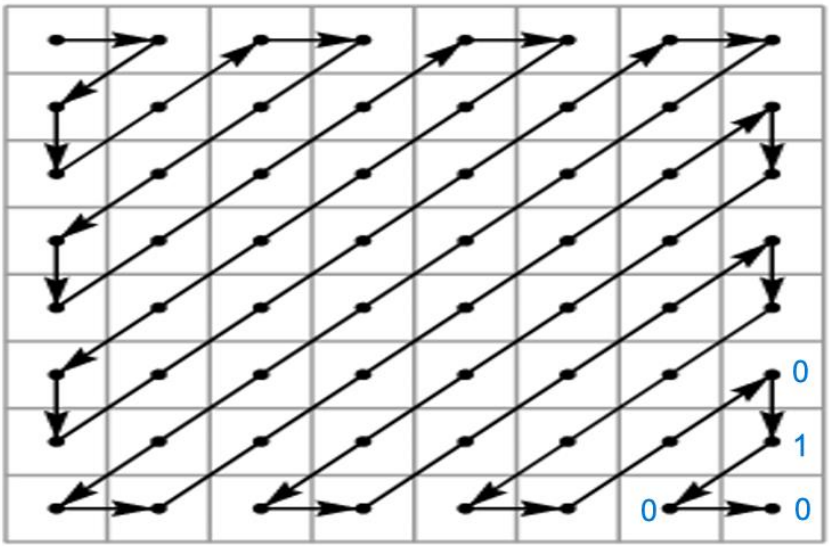
Sau quá trình biển đổi DCT và lượng tử hoá thành công các khối ma trận 8x8 trên, lúc này ta có thể thực hiện giấu các bit thông điệp vào trong các khối ma trận 8x8 trên.

Trước tiên, sinh viên cần chuyển đổi thông điệp cần giấu là *"ATT"* từ dạng ASCII sang một chuỗi 24 bit nhị phân *(sinh viên có thể sử dụng các tool chuyển đổi online).*

Khi đã xác định được chuỗi 24 bit thông điệp ở dạng nhị phân, do trước đó ta tiến hành trích xuất 6 khối ma trận 8x8, do vậy với mỗi khối ta sẽ thực hiện giấu *4 bit* nhị phân tương ứng.

Trong bài thực hành này, mỗi ma trận sẽ được thay thế *4 bit* cuối theo chiều thuật toán ma trận zigzag bằng *x* bit nhị phân của thông điệp giấu.

Ví dụ nếu ta có 4 bit cần giấu là *0100*, do đó theo chiều ma trận zigzag như bên dưới ta sẽ cần thực hiện giấu theo chiều như bên dưới:



Sinh viên hãy tiến hành chỉnh sửa thay thế 4 bit cuối theo chiều thuật toán zigzag thành 4 bit nhị phân cần giấu, lần lượt theo trình tự tương ứng các khối ma trận đã trích xuất.

3.4 Phục hồi ảnh

Để có thể thực hiện phục hồi lại ảnh, ta cần biến đổi lại các khối ma trận 8x8 mà trước đó ta đã thực hiện các bước: biến đổi DCT, lượng tử hóa và giấu các bit thông điệp về thành lại dạng các khối ma trận có hệ số điểm ảnh trong khoảng [0,255].

Trên terminal, ta gõ các lệnh:

python3 dct_transform/create_new_matrix.py
extracted_matrix/matrix_8x8_vitri_<i>_<j>.txt

Chương trình *create_new_matrix.py* trong thư mục *dct_transform* tiến hành lần lượt nhân khối ma trận 8x8 với ma trận lượng tử hóa, và sau đó thực hiện biến đổi DCT ngược (IDCT) để thu được khối ma trận mới. Nếu ta so sánh 2 khối ma trận điểm ảnh ban đầu và mới thì sẽ thấy có rất ít thông tin bị thay đổi giữa chúng.

Sau quá trình trên, tiếp đó ta sẽ thay thế các khối ma trận này lại vào trong ma trận điểm ảnh tổng *image_matrix.txt* bằng chương trình *rewrite_matrix.py*.

Trên terminal, ta gõ các lệnh:

python3 rewrite_matrix.py image_matrix.txt <i> <j>
extracted_matrix/matrix_8x8_vitri_<i>_<j>.txt

File *image_matrix.txt* được thay thế thành công, sinh viên có thể *nano* để kiểm tra.

Thực hiện phục hồi lại ảnh từ ma trận điểm ảnh mới *image_matrix.txt* mà đã được giấu thông điệp. Trên terminal, ta gõ lệnh:

python3 recovery_image.py image_matrix.txt

Kết quả thu được ảnh *recovered_image.png* đã được giấu tin, sinh viên cũng có thể sử dụng lệnh *fim recovered_image.png* để xem ảnh.

3.5 Kết thúc bài lab

Trên terminal, gõ:

stoplab

Khi bài lab kết thúc, một tệp lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới.

Sinh viên cần nộp file *.lab* để chấm điểm.

Để kiểm tra kết quả khi trong khi làm bài thực hành sử dụng lệnh:

checkwork

Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, sử dụng lệnh:

labtainer –r stego_dct_code