Hướng Dẫn Thực Hành: Watermark Âm Thanh Bằng Phương Pháp Echo Hiding

Tháng 5, 2025

1 Giới Thiệu

Bài thực hành này hướng dẫn sinh viên triển khai **watermark âm thanh**, một kỹ thuật nhúng thông tin ẩn (watermark) vào tín hiệu âm thanh để bảo vệ bản quyền hoặc xác thực nguồn gốc. Phương pháp được sử dụng là **echo hiding**, nhúng watermark bằng cách thêm các tiếng vọng với độ trễ cụ thể. Bài thực hành được thực hiện trong môi trường **Labtainer**, một nền tảng an toàn hỗ trợ chấm điểm tự động.

1.1 Mục Tiêu

- Hiểu các nguyên tắc của watermark âm thanh và phương pháp echo hiding.
- Thực hiện các bước: tạo watermark, nhúng watermark, phát hiện watermark và đánh giá hiệu suất.
- Tích lũy kinh nghiệm về xử lý tín hiệu số, phân tích cepstrum, và các chỉ số như Tỷ Lệ Lỗi Bit (BER) và Tỷ Số Tín Hiệu Trên Nhiễu (SNR).
- Làm chủ lập trình Python trong môi trường Labtainer.

1.2 Cấu Trúc Bài Thực Hành

Bài thực hành gồm 5 nhiệm vụ, mỗi nhiệm vụ liên quan đến một hoặc nhiều kịch bản Python:

1. **Nhiệm Vụ 1**: Tạo watermark và khóa bí mật (`task1_generate_keys.py`).
2. **Nhiệm Vụ 2**: Tính toán tham số nhúng (`task2_compute_params.py`).
3. **Nhiệm Vụ 3**: Nhúng watermark bằng ba phương pháp:
    - Tiếng vọng dương (`task3_embed_positive.py`).
    - Tiếng vọng dương + âm (`task3_embed_negative.py`).
    - Tiếng vọng dương + tiến (`task3_embed_forward.py`).

4. **Nhiệm Vụ 4**: Phát hiện watermark (`task4_detect.py`).
5. **Nhiệm Vụ 5**: Đánh giá hiệu suất (BER và SNR) (`task5_evaluate.py`).

2 Kiến Thức Nền

2.1 Watermark Âm Thanh

Watermark âm thanh nhúng một chuỗi bit (watermark) vào tín hiệu âm thanh sao cho:

- **Tính không nhận biết được**: Watermark không làm giảm chất lượng âm thanh một cách đáng chú ý.
- **Tính bền vững**: Watermark có thể được phát hiện dù tín hiệu bị nén hoặc có nhiễu.
- **Tính bảo mật**: Chỉ những người có khóa bí mật mới phát hiện được watermark.

2.2 Phương Pháp Echo Hiding

Phương pháp echo hiding nhúng watermark bằng cách thêm tiếng vọng vào tín hiệu âm thanh. Mỗi bit (0 hoặc 1) được biểu diễn bằng một độ trễ cụ thể:

- Bit 1: Tiếng vọng với độ trễ 100 mẫu (cho khóa bí mật = 1) hoặc 120 mẫu (cho khóa bí mật = 0).
- Bit 0: Tiếng vọng với độ trễ 110 mẫu (cho khóa bí mật = 1) hoặc 130 mẫu (cho khóa bí mật = 0).

Bài thực hành sử dụng ba loại tiếng vọng:

- **Tiếng vọng dương**: Thêm tiếng vọng với biên độ dương.
- **Tiếng vọng dương + âm**: Thêm cả tiếng vọng dương và âm để tăng tính bền vững.
- **Tiếng vọng dương + tiến**: Thêm tiếng vọng dương và tiếng vọng dịch tiến.

Quá trình nhúng:

1. Chia tín hiệu âm thanh thành các khung (4096 mẫu).
2. Thêm tiếng vọng với độ trễ phù hợp vào mỗi khung dựa trên bit watermark và khóa bí mật.
3. Áp dụng cửa sổ Hann để giảm hiện tượng méo ở ranh giới khung.

2.3 Phân Tích Cepstrum

Phát hiện watermark sử dụng **cepstrum**, được tính như sau:

$$\text{ceps}(n) = \text{IFFT}\left(\log(|\text{FFT}(x)|^2 + \epsilon)\right)$$

trong đó $\epsilon = 0.00001$ để tránh log(0). Tiếng vọng xuất hiện dưới dạng đỉnh trong cepstrum tại các vị trí độ trễ. So sánh giá trị cepstrum tại các độ trễ để xác định bit:

- Đối với `signal1` và `signal3`: So sánh trực tiếp các đỉnh (ví dụ, ceps[100] so với ceps[110]).
- Đối với `signal2`: So sánh hiệu số (ví dụ, ceps[100] - ceps[104] so với ceps[110] - ceps[114]).

2.4 Mã Hóa Lặp

Mỗi bit watermark được lặp lại 3 lần để tăng tính bền vững. Trong quá trình phát hiện, các bit lặp được tổng hợp:

$$\text{Bit được phát hiện} = \begin{cases} 1 & \text{nếu trung bình} \geq 0.5 \\ 0 & \text{nếu trung bình} < 0.5 \end{cases}$$

2.5 Chỉ Số Đánh Giá

- **Tỷ Lệ Lỗi Bit (BER)**: Phần trăm các bit sai:

  $$\text{BER} = \frac{\text{Số bit sai}}{\text{Tổng số bit}} \times 100\%$$

  BER thấp hơn là tốt hơn.

- **Tỷ Số Tín Hiệu Trên Nhiễu (SNR)**: Tỷ số tín hiệu trên nhiễu:

  $$\text{SNR} = 10 \log_{10} \left(\frac{\sum x^2}{\sum (x - y)^2} \right) \text{ dB}$$

  (x: tín hiệu gốc, y: tín hiệu đã nhúng watermark). SNR cao hơn là tốt hơn.

3 Hướng Dẫn

3.1 Yêu Cầu

- Môi trường Labtainer đã được cài đặt.
- File `bass_half.wav` trong thư mục `/home/student`.
- Python 3, `numpy`, `scipy` đã được cài đặt trong container.
- Trình xem PDF (ví dụ, `evince`) để mở `huong_dan.pdf`.

3.2 Khởi Động Labtainer

1. Khởi động bài thực hành:

   labtainer watermark_echo_hiding -r

2. Thư mục `/home/student` chứa:
   - File âm thanh: `bass_half.wav`.
   - Kịch bản: `task1_generate_keys.py`, `task2_compute_params.py`, `task3_embed_positive.py`, `task3_embed_negative.py`, `task3_embed_forward.py`, `task4_detect.py`, `task5_evaluate.py`.
   - Hướng dẫn: `watermark_echo_hiding.pdf`.

3. Mở hướng dẫn:

   xdg-open watermark_echo_hiding.pdf

3.3 Chi Tiết Nhiệm Vụ

3.3.1 Nhiệm Vụ 1: Tạo Watermark và Khóa Bí Mật

Mô tả: Tạo một watermark ngẫu nhiên 40 bit và khóa bí mật, mở rộng thành 120 bit (lặp lại 3 lần mỗi bit) để tăng tính bền vững.

Đầu ra:

- `watermark_ori.dat`: Watermark gốc (40 bit).
- `watermark_extended.dat`: Watermark mở rộng (120 bit).
- `secret_key.dat`: Khóa bí mật mở rộng (120 bit).

Hướng dẫn:

```
python task1_generate_keys.py --watermark_original_file
   watermark_ori.dat --watermark_extended_file
   watermark_extended.dat --secret_key_file secret_key.dat
```

Kiểm tra:

- Mở `watermark_ori.dat` (40 dòng, mỗi dòng là 0 hoặc 1).
- Mở `watermark_extended.dat` và `secret_key.dat` (120 dòng).

Kết quả học tập:

- Tạo chuỗi bit ngẫu nhiên.
- Hiểu về mã hóa lặp trong watermark.

3.3.2 Nhiệm Vụ 2: Tính Toán Tham Số Nhúng

Mô tả: Đọc `bass_half.wav`, tính toán độ dịch khung và số bit có thể nhúng, sau đó lưu tham số.

Đầu vào: `bass_half.wav`.

Đầu ra: `embed_params.dat` (chứa `frame_shift`, `embed_nbit`, `effective_nbit`).

Hướng dẫn:

```
python task2_compute_params.py --host_signal_file bass_half.wav
   --output_file embed_params.dat
```

Kiểm tra:

- Mở `embed_params.dat` để xem các tham số.
- Đảm bảo `effective_nbit` không vượt quá 40.

Kết quả học tập:

- Đọc file WAV và phân khung.
- Tính toán tham số nhúng dựa trên độ dài tín hiệu.

3.3.3 Nhiệm Vụ 3: Nhúng Watermark Bằng Phương Pháp Echo Hiding

Mô tả: Nhúng watermark vào `bass_half.wav` bằng ba phương pháp echo hiding:

- Tiếng vọng dương: Thêm tiếng vọng dương.
- Tiếng vọng dương + âm: Thêm tiếng vọng dương và âm.
- Tiếng vọng dương + tiến: Thêm tiếng vọng dương và tiến.

Đầu vào: `bass_half.wav`, `watermark_extended.dat`, `secret_key.dat`, `embed_params.dat`.

Đầu ra:

- `wmed_signal1.wav` (tiếng vọng dương).
- `wmed_signal2.wav` (tiếng vọng dương + âm).
- `wmed_signal3.wav` (tiếng vọng dương + tiến).

Hướng dẫn:

```
python task3_embed_positive.py --host_signal_file bass_half.wav
   --watermark_extended_file watermark_extended.dat
   --secret_key_file secret_key.dat --params_file embed_params.dat
   --output_file wmed_signal1.wav
python task3_embed_negative.py --host_signal_file bass_half.wav
   --watermark_extended_file watermark_extended.dat
   --secret_key_file secret_key.dat --params_file embed_params.dat
   --output_file wmed_signal2.wav
python task3_embed_forward.py --host_signal_file bass_half.wav
   --watermark_extended_file watermark_extended.dat
   --secret_key_file secret_key.dat --params_file embed_params.dat
   --output_file wmed_signal3.wav
```

Kiểm tra:

- Nghe `wmed_signal1.wav`, `wmed_signal2.wav`, `wmed_signal3.wav` để kiểm tra chất lượng âm thanh.
- Đảm bảo các file có cùng độ dài và định dạng WAV như `bass_half.wav`.

Kết quả học tập:

- Kỹ thuật echo hiding.
- Xử lý tín hiệu với khung và cửa sổ Hann.
- So sánh các loại tiếng vọng (dương, âm, tiến).

3.3.4 Nhiệm Vụ 4: Phát Hiện Watermark

Mô tả: Phát hiện watermark từ các file âm thanh đã nhúng watermark bằng phân tích cepstrum. Kịch bản hỗ trợ cả ba loại tín hiệu (`signal1`, `signal2`, `signal3`).

Đầu vào: `wmed_signalX.wav`, `secret_key.dat`, `embed_params.dat`.
Đầu ra:

- `detected_bits1.dat` (cho `wmed_signal1.wav`).
- `detected_bits2.dat` (cho `wmed_signal2.wav`).
- `detected_bits3.dat` (cho `wmed_signal3.wav`).

Hướng dẫn:

```
1 python task4_detect.py --watermark_signal_file wmed_signal1.wav
    --secret_key_file secret_key.dat --params_file embed_params.dat
    --output_file detected_bits1.dat --signal_type signal1
2 python task4_detect.py --watermark_signal_file wmed_signal2.wav
    --secret_key_file secret_key.dat --params_file embed_params.dat
    --output_file detected_bits2.dat --signal_type signal2
3 python task4_detect.py --watermark_signal_file wmed_signal3.wav
    --secret_key_file secret_key.dat --params_file embed_params.dat
    --output_file detected_bits3.dat --signal_type signal3
```

Kiểm tra:

- Mở `detected_bitsX.dat` (120 dòng, mỗi dòng là 0 hoặc 1).
- So sánh với `watermark_extended.dat` để kiểm tra sơ bộ.

Kết quả học tập:

- Phân tích cepstrum để phát hiện tiếng vọng.
- Xử lý tín hiệu FFT/IFFT.
- Ảnh hưởng của tiếng vọng âm đến việc phát hiện.

3.3.5 Nhiệm Vụ 5: Đánh Giá Hiệu Suất

Mô tả: Tính toán BER và SNR để đánh giá chất lượng nhúng và phát hiện cho mỗi file âm thanh.

Đầu vào: `bass_half.wav`, `wmed_signalX.wav`, `watermark_ori.dat`, `detected_bitsX.dat`, `embed_params.dat`.

Đầu ra:

- `results1.txt` (cho `wmed_signal1.wav`).
- `results2.txt` (cho `wmed_signal2.wav`).
- `results3.txt` (cho `wmed_signal3.wav`).

Hướng dẫn:

```
1 python task5_evaluate.py --host_signal_file bass_half.wav
    --watermark_signal_file wmed_signal1.wav
    --watermark_original_file watermark_ori.dat --detected_bits_file
    detected_bits1.dat --params_file embed_params.dat
2 python task5_evaluate.py --host_signal_file bass_half.wav
    --watermark_signal_file wmed_signal2.wav
    --watermark_original_file watermark_ori.dat --detected_bits_file
    detected_bits2.dat --params_file embed_params.dat
3 python task5_evaluate.py --host_signal_file bass_half.wav
    --watermark_signal_file wmed_signal3.wav
    --watermark_original_file watermark_ori.dat --detected_bits_file
    detected_bits3.dat --params_file embed_params.dat
```

Kiểm tra:

- Mở `resultsX.txt` để xem BER và SNR.
- So sánh BER (phương pháp nào có BER thấp nhất?) và SNR (phương pháp nào giữ chất lượng âm thanh tốt nhất?).

Kết quả học tập:

- Tính toán BER và SNR.
- Đánh giá hiệu suất phương pháp nhúng.
- Hiểu sự đánh đổi giữa chất lượng âm thanh và độ bền vững của watermark.

4 Lưu Ý

- Thực hiện các nhiệm vụ theo thứ tự (từ 1 đến 5).
- Đảm bảo các file đầu vào (`bass_half.wav`, `watermark_extended.dat`, v.v.) tồn tại trước khi chạy.
- Nếu kịch bản thất bại, kiểm tra:
  - Tên và định dạng file đúng.

- Đầu ra của kịch bản khớp với yêu cầu.

- Lưu các file đầu ra (`wmed_signalX.wav`, `resultsX.txt`) để phân tích sau này.

5 Câu Hỏi Thảo Luận

1. Tại sao mỗi bit watermark được lặp lại ba lần (mã hóa lặp)? Điều này ảnh hưởng thế nào đến BER?
2. Nghe `wmed_signal1.wav`, `wmed_signal2.wav`, `wmed_signal3.wav`. Bạn có nhận thấy sự khác biệt về chất lượng âm thanh so với `bass_half.wav` không?
3. Dựa trên `results1.txt`, `results2.txt`, `results3.txt`, phương pháp nào (`signal1`, `signal2`, `signal3`) có BER thấp nhất và SNR cao nhất? Tại sao?
4. Nếu âm thanh bị nén (ví dụ, sang MP3), điều gì có thể xảy ra với watermark? Phương pháp nào có khả năng bền vững hơn?

6 Thí Nghiệm Mở Rộng

Để hiểu sâu hơn, hãy thử:

1. Sửa đổi `task3_embed_positive.py` (hoặc tương tự) để thay đổi `CONTROL_STRENGTH` (ví dụ, 0.1 hoặc 0.3). Chạy lại Nhiệm Vụ 3, 4, 5 và so sánh BER/SNR.
2. Tắt mã hóa lặp bằng cách đặt `REP_CODE=False` trong `task1_generate_keys.py` và `task2_compute_params.py`. So sánh BER khi bật và tắt mã hóa lặp.
3. Thay đổi `FRAME_LENGTH` (ví dụ, 2048 hoặc 8192) trong các file Nhiệm Vụ 2 và 3. Quan sát ảnh hưởng đến `embed_nbit` và chất lượng âm thanh.

7 Kết Luận

Bài thực hành này cung cấp kiến thức thực tiễn về watermark âm thanh, xử lý tín hiệu số và lập trình Python. Sau khi hoàn thành, sinh viên sẽ:

- Hiểu về nhúng và phát hiện watermark bằng phương pháp echo hiding.
- Làm chủ phân tích cepstrum và tính toán BER/SNR.
- Làm quen với Labtainer và quy trình bài thực hành tự động.

Chúc bạn thực hành thành công và thú vị!