ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

LỚP TRÍ TUỆ NHÂN TẠO 2023 - 23TNT1

Báo cáo Đồ án #1

Đề tài: Vitual Try-On

Môn học: Thực hành - Giới thiệu ngành Trí tuệ nhân tạo

Sinh viên thực hiện:
Đinh Đức Anh Khoa (23122001)
Nguyễn Đình Hà Dương (23122002)
Nguyễn Lê Hoàng Trung (23122004)
Đinh Đức Tài (23122013)

Giáo viên hướng dẫn:
CN. Nguyễn Bảo Long

Ngày 11 tháng 12 năm 2023

Mục lục

1 Giới thiệu

Đây là bài báo cáo Đồ án #1, môn Thực hành Giới thiệu ngành Trí tuệ Nhân tạo.
Đồ án #1 được thực hiện bởi nhóm các thành viên:

- Đinh Đức Anh Khoa (23122001)
- Nguyễn Lê Hoàng Trung (23122002)
- Nguyễn Đình Hà Dương (23122004)
- Đinh Đức Tài (23122013)

2 Phân công nhiệm vụ

Sau đây là bảng phân công nhiệm vụ cho từng thành viên :

Họ và tên	MSSV	Chức vụ	Nhiệm vụ
Đinh Đức Anh Khoa	23122001	Nhóm trưởng	- Code Python lấy mask của ảnh quần áo - Code thực thi và demo Human Parsing
Nguyễn Đình Hà Dương	23122002	Thành viên	- Demo rút trích mask, Pose Estimation - Làm slide trình chiếu
Nguyễn Lê Hoàng Trung	23122004	Thành viên	- Chỉnh sửa và chú thích video - Cài đặt môi trường thực thi, demo
Đinh Đức Tài	23122013	Thành viên	- Nghiên cứu tài liệu và chọn các mô hình - Mô tả các khái niệm, cách mô hình hoạt động - Viết báo cáo đồ án

3 Tổng quan công việc

3.1 Tuần 1

Nội dung công việc thực hiện trong Tuần 1:

- Thiết lập môi trường thực thi
- Chuẩn bị ảnh đầu vào
- Tạo mask cho ảnh quần áo
- Demo

3.2 Tuần 2

Nội dung công việc thực hiện trong Tuần 2:

- Tìm phân đoạn bố cục ảnh người (human parsing)
- Rút trích đặc trưng dáng người (pose estimation)
- Demo

3.3 Tuần 3

Nội dung công việc thực hiện trong Tuần 3:

- Khởi chạy mô hình Try-on
- Giải thích các mô hình pre-trained
- Demo

4 Tuần 1

4.1 Thiết lập môi trường thực thi

Giải thích các khái niệm:

Conda:

- Conda là một hệ thống quản lý gói và quản lý môi trường mã nguồn mở để cài đặt nhiều phiên bản của các gói phần mềm và các phần phụ thuộc, có thể linh hoạt chuyển đổi giữa chúng. Conda hoạt động trên hệ điều hành Linux, OS X và Windows, và được tạo ra cho các chương trình Python nhưng cũng có thể đóng gói và phân phối các phần mềm khác.[1]

Miniconda:

- Miniconda là một công cụ cài đặt nhỏ gọn và miễn phí cho conda. Đây là phiên bản khởi động nhỏ của Anaconda chỉ bao gồm conda, Python, các gói phụ thuộc của chúng cùng với một số gói hữu ích khác (như pip, zlib và một số gói khác). [2]

Môi trường thực thi:

- Môi trường thực thi là một môi trường ảo được tạo ra bởi công cụ quản lý môi trường (Miniconda). Môi trường thực thi cho phép người dùng tạo, quản lý và chia sẻ các môi trường cài đặt độc lập của Python và các gói phụ thuộc liên quan [3]
- Môi trường vừa tạo gồm những thành phần:
  1. Repository `VITON-HD` [4] và các pre-trained networks[5]
  2. `Python` phiên bản 3.8
  3. `PyTorch` phiên bản 1.6.0 trở lên và `torchvision`
  4. `CUDA Toolkit` phiên bản 11.8, đã cài biến môi trường
  5. Gói `pytorch` từ kho lưu trữ `conda`
  6. Gói `opencv-python` và `torchgeometry` của `Python`
- Lý do phải tạo môi trường thực thi: Môi trường thực thi cung cấp một không gian làm việc cô lập trong đó ta có thể cài đặt các phiên bản khác nhau của Python và các gói phụ thuộc mà không ảnh hưởng đến các môi trường khác. Điều này rất hữu ích khi làm việc trên nhiều dự án có yêu cầu khác nhau hoặc khi cần đảm bảo sự nhất quán và độ tin cậy của môi trường cài đặt.

4.2 Chuẩn bị hình ảnh đầu vào

Từ `dữ liệu ảnh` [6] (ở độ phân giải 1024x768) được cung cấp, sử dụng ảnh quần áo, phụ kiện `C` và ảnh con người `P` để tạo các thành phần:

- Lấy mask của C, thu được m_C (Tuần 1)
- Phân đoạn bố cục người (human parsing) của P, thu được s_P (Tuần 2)
- Rút trích đặc trưng dáng (pose) cho P, thu được p_P (Tuần 2)

4.3 Tạo mask cho ảnh quần áo

4.3.1 Công cụ rút trích mask của ảnh

- Sử dụng code Python
- Thư viện OpenCV để xử lí hình ảnh

4.3.2 Cách thức hoạt động của việc tạo mask

1. Đọc ảnh từ thư mục "image"
2. Chuyển đổi ảnh sang không gian màu grayscale
3. Tạo một ma trận mask ban đầu với kích thước tương tự ảnh grayscale, tất cả các giá trị trong mask ban đầu đều là màu trắng (255)
4. Sử dụng BFS để tìm phần nền
   - Chọn điểm (0,0) trên ảnh làm điểm bắt đầu
   - Duyệt qua các điểm ảnh lân cận và đánh dấu các điểm thuộc phần nền trong mask bằng cách đặt giá trị tương ứng là màu đen (0)
   - Tiếp tục BFS cho đến khi không còn điểm ảnh nào được duyệt
5. Tạo một bản sao của mask và đặt tất cả các giá trị trong bản sao đó là màu đen (0)
6. Sử dụng BFS lần thứ hai để giảm vỡ mask
   - Chọn một điểm nằm ở giữa ảnh làm điểm bắt đầu
   - Duyệt qua các điểm ảnh hàng xóm và đánh dấu các điểm thuộc phần mask trong mask bằng cách đặt giá trị tương ứng là màu trắng (255)
   - Tiếp tục BFS cho đến khi không còn điểm ảnh nào được duyệt
7. Xuất mask cuối cùng ra thư mục "mask"

4.3.3 Code

```
import cv2
import os
dx = [1, -1, 0, 0]
dy = [0, 0, 1, -1]

def bfs_black(x, y, a, mask):
    mask[x][y] = 0
    queue = [(x, y)]

    while len(queue):
        current_x, current_y = queue[len(queue) - 1]
        queue.pop()

        for i in range(4):
```

```
            nx = dx[i] + current_x
            ny = dy[i] + current_y

            if nx < 0 or ny < 0 or nx >= len(a) or ny >= len(a[0]) or mask[nx][ny] == 0 or a[nx][ny] < 246:
                continue

            mask[nx][ny] = 0
            queue.append((nx, ny))

    return mask

def bfs_white(x, y, a, mask):
    mask[x][y] = 255
    queue = [(x, y)]

    while len(queue):
        current_x, current_y = queue[len(queue) - 1]
        queue.pop()

        for i in range(4):
            nx = dx[i] + current_x
            ny = dy[i] + current_y

            if nx < 0 or ny < 0 or nx >= len(a) or ny >= len(a[0]) or mask[nx][ny] == 255 or a[nx][ny] == 0:
                continue

            mask[nx][ny] = 255
            queue.append((nx, ny))

    return mask

def create_mask(image):

    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    mask = gray_image.copy()
    for x in range(0, len(mask)):
        for y in range(0, len(mask[x])):
            mask[x][y] = 255
    bfs_black(0, 0, gray_image, mask)

    mask2 = mask.copy()
    for x in range(0, len(mask2)):
        for y in range(0, len(mask2[x])):
            mask2[x][y] = 0
    bfs_white(len(mask2) // 2, len(mask2[0]) // 2, mask, mask2)

    return cv2.medianBlur(mask2, 5)

def main():
    for filename in os.listdir("image"):

        image = cv2.imread(os.path.join("image", filename))
```

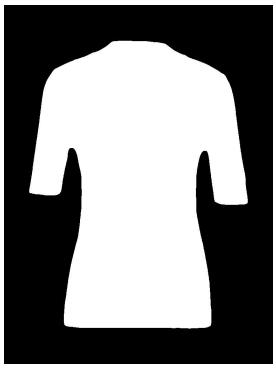
```
68
69          mask = create_mask(image)
70          print('complete masking ' + filename)
71
72          cv2.imwrite(os.path.join("mask", filename[:-4] + "_mask.jpg"), mask)
73
74  main()
```

4.4 Demo

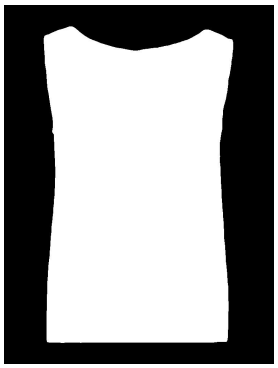
Link demo Đồ án: https://youtu.be/jSBkMBHlwrU

4.5 So sánh

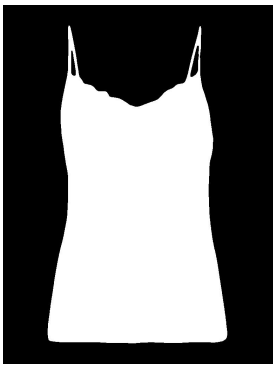
Sau đây là kết quả so sánh cloth-mask giữa dữ liệu của tác giả (ảnh gốc) và kết quả rút trích mask của nhóm (Mask):



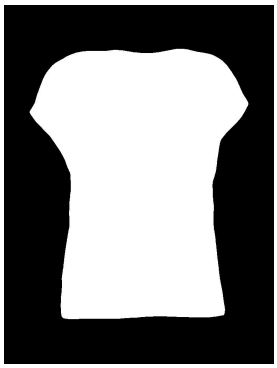
Hình 1: Ảnh gốc 1



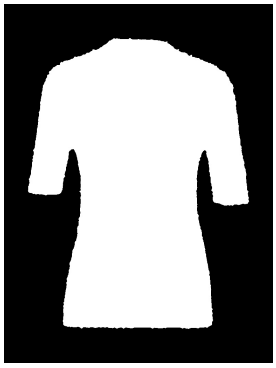
Hình 2: Ảnh gốc 2



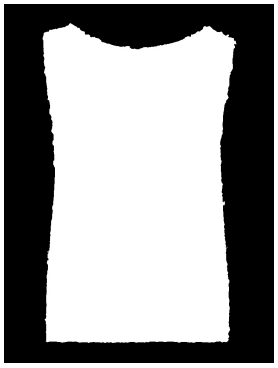
Hình 3: Ảnh gốc 3



Hình 4: Ảnh gốc 4

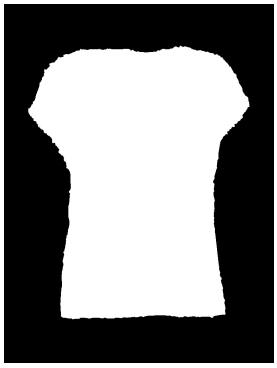


Hình 5: Mask 1



Hình 6: Mask 2

Hình 7: Mask 3



Hình 8: Mask 4

5 Tuần 2

5.1 Các khái niệm

5.1.1 Huấn luyện mô hình học máy

Quá trình huấn luyện mô hình học máy là quá trình cung cấp dữ liệu đầu vào và đầu ra cho mô hình, để mô hình có thể học từ dữ liệu và tìm ra một hàm ánh xạ từ dữ liệu đầu vào sang đầu ra tương ứng. Mô hình được huấn luyện thông qua việc điều chỉnh các tham số (trọng số) của nó dựa trên sự khác biệt giữa đầu ra thực tế và đầu ra được dự đoán bởi mô hình.

Input, Output: Input (đầu vào) của mô hình học máy là dữ liệu đầu vào, có thể là ảnh, văn bản, âm thanh, hoặc các dạng dữ liệu khác. Output (đầu ra) của mô hình là dự đoán mà mô hình tạo ra dựa trên đầu vào. Ví dụ, nếu mô hình được huấn luyện để nhận dạng hình ảnh, đầu vào sẽ là một hình ảnh và đầu ra sẽ là nhãn xác định loại đối tượng trong hình ảnh đó.

Quá trình huấn luyện mô hình học máy thường diễn ra qua các bước sau:

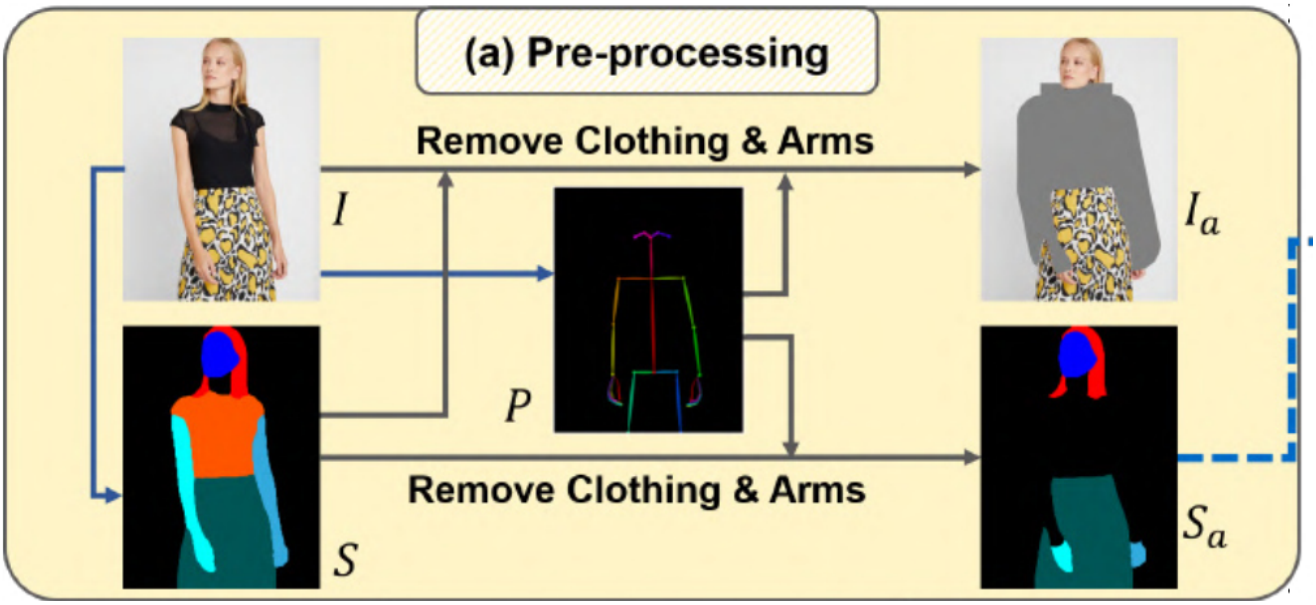
1. Chuẩn bị dữ liệu: Dữ liệu huấn luyện được chuẩn bị và tiền xử lý để đảm bảo độ chuẩn xác và tính đại diện của nó.
2. Xây dựng mô hình: Mô hình học máy được xây dựng với kiến trúc và các tham số ban đầu.
3. Định nghĩa hàm mất mát: Một hàm mất mát được chọn để đo lường sai khác giữa đầu ra dự đoán và đầu ra thực tế.
4. Tối ưu hóa tham số: Mô hình sử dụng một thuật toán tối ưu (như gradient descent) để điều chỉnh các tham số (trọng số) của mô hình để giảm thiểu hàm mất mát.
5. Đánh giá mô hình: Mô hình được đánh giá thông qua việc sử dụng dữ liệu kiểm tra hoặc dữ liệu đánh giá độc lập để đo lường hiệu suất của nó.

5.1.2 Pre-trained model

Pre-trained model là mô hình đã được huấn luyện với một lượng dữ liệu lớn, có khả năng hoàn thành được nhiệm vụ cụ thể. Pre-trained model có thể được sử dụng nguyên trạng hoặc tùy chỉnh để phù hợp với yêu cầu sử dụng. [7]

5.2 Tổng quan về Human parsing và Pose estimation

5.2.1 Mục đích và phương pháp



Hình 9: Quá trình pre-processing của VITON-HD

Lý do VITON-HD cần phân đoạn bố cục người (segmentation map) và đặc trưng dáng người (pose map):

- Nhằm tạo hình ảnh không phụ thuộc vào quần áo $\mathtt{I_a}$ và một bản đồ phân đoạn không phụ thuộc vào quần áo. $\mathtt{S_a}$.
- Bản đồ phân đoạn S được sử dụng để loại bỏ vùng quần áo cần thay thế và bảo tồn phần còn lại của hình ảnh.
- Bản đồ dáng người P được sử dụng để loại bỏ cánh tay, nhưng không loại bỏ tay, vì chúng khó tái tạo.
- Dựa trên S và P, tạo ra hình ảnh không phụ thuộc vào quần áo $\mathtt{I_a}$ và bản đồ phân đoạn không phụ thuộc vào quần áo $\mathtt{S_a}$, cho phép mô hình loại bỏ thông tin quần áo gốc một cách toàn diện và bảo tồn phần còn lại của hình ảnh.
- Các thành phần $\mathtt{I_a}$, $\mathtt{S_a}$, S và P là một trong các input (đầu vào) cho 3 pre-trained model của VITON-HD.

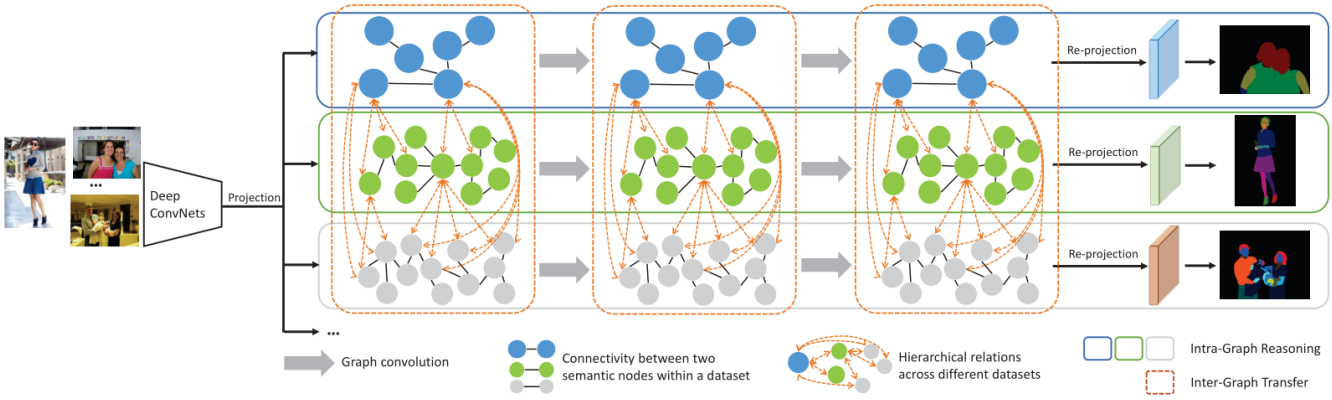
Công cụ tạo bản đồ phân đoạn bố cục người và bản đồ dáng người:

- Dùng mô hình `Graphonomy: Universal Human Parsing via Graph Transfer Learning` [8] để tạo bản đồ phân đoạn bố cục người (segmentation map - `S`)
- Dùng mô hình `OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields` [9] để tạo bản đồ dáng người (pose map - `P`)

5.3 Phân đoạn bố cục người (Human Parsing)

Mô hình được sử dụng: Graphonomy: Universal Human Parsing via Graph Transfer Learning [8]

5.3.1 Graphonomy



Hình 10: Cách hoạt động của mô hình Graphonomy: Sử dụng 2 module Intra-Graph Reasoning (tính toán trong đồ thị nội bộ) và Inter-Graph Transfer (truyền thông tin giữa các đồ thị) tạo ra biểu diễn đồ thị ngữ nghĩa cấp cao từ các tập dữ liệu khác nhau, cung cấp kết quả tốt nhất cho các tác vụ phân tích con người ở nhiều cấp độ khác nhau.

Quy trình tổng thể mô hình Graphonomy:

- Graphonomy là một mô hình phân tích con người thông qua việc học từ một tập hợp các tác vụ phân tích con người khác nhau. Mô hình này sử dụng học chuyển đổi đồ thị để mã hoá cấu trúc ngữ nghĩa của các nhãn và truyền thông tin ngữ nghĩa liên quan.
- Để làm điều này, Graphonomy sử dụng hai module chính: Intra-Graph Reasoning (tính toán trong đồ thị nội bộ) và Inter-Graph Transfer (truyền thông tin giữa các đồ thị).
- Module Intra-Graph Reasoning được sử dụng để cải tiến biểu diễn đồ thị bên trong cùng một tập dữ liệu. Ban đầu, các đặc trưng hình ảnh được chiếu vào một đồ thị, trong đó các pixel có đặc trưng tương tự được gán cho cùng một đỉnh ngữ nghĩa. Mô hình sử dụng ma trận kề để mã hoá các mối quan hệ ngữ nghĩa giữa các đỉnh, trong đó các mối quan hệ này được ràng buộc bởi cấu trúc cơ thể con người. Sau đó, thông tin được truyền từ các đỉnh đến các đỉnh khác thông qua các phép tính đồ thị, và sau đó các đỉnh được cập nhật để tạo ra các bản đồ đặc trưng hình ảnh phân loại pixel.
- Module Inter-Graph Transfer được sử dụng để truyền thông tin ngữ nghĩa giữa các đồ thị khác nhau. Nó cho phép trích xuất các ngữ nghĩa liên quan từ đồ thị trong một tác vụ/domain sang đồ thị trong tác vụ/domain khác. Để tăng khả năng truyền thông tin giữa các đồ thị, mô hình sử dụng các phụ thuộc chuyển đổi đồ thị khác nhau, bao gồm sự tương đồng về đặc trưng và sự tương đồng ngữ nghĩa được mô tả bằng kiến thức ngôn ngữ.

- Bằng cách kết hợp Intra-Graph Reasoning và Inter-Graph Transfer, Graphonomy kết hợp và tạo ra biểu diễn đồ thị ngữ nghĩa cấp cao từ các tập dữ liệu khác nhau, cung cấp kết quả tốt nhất cho các tác vụ phân tích con người ở nhiều cấp độ khác nhau.

5.3.2 Input, Output

Input (dữ liệu đầu vào) và Output (dữ liệu đầu ra) của tác vụ human parsing:

- Input: Ảnh người, độ phân giải 1024x768
- Output: Ảnh bản đồ phân đoạn bố cục người (segmentation map `S`), độ phân giải 1024x768

5.3.3 Cài đặt và hướng dẫn sử dụng

1. Môi trường thực thi và pre-trained network
   Ta thêm các thành phần mới dựa trên môi trường thực thi đã tạo ở Tuần 1. Trong môi trường thực thi này, ta có:
   - Repository `Graphonomy`
   - Pre-trained network : CIHP trained model [10] hoặc Pascal-Person-Part trained model [11] và lưu trữ trong thư mục Graphonomy
   - `Python` phiên bản 3.8
   - `CUDA Toolkit` phiên bản 11.8 [12], đã cài biến môi trường
   - Các thư viện và gói sau: `pytorch`, `tensorboardX`, `opencv-python`, `scipy`, `numpy`, `matplotlib`, `networkx`, `torchvision`, `torchaudio`, `torchgeometry`
2. Phương pháp thực thi
   - Khi sử dụng ảnh làm đầu vào (input) có độ phân giải cao, `Graphonomy` sẽ rất tốn dung lượng RAM.
   - Phương pháp được đề xuất: Giảm kích thước hình ảnh đầu vào và đưa cho mô hình `Graphonomy` xử lí. Sau đó đưa kích thước hình ảnh trở lại với độ phân giải ban đầu (1024x768)
   - Kết hợp code giảm kích thước hình ảnh và các câu lệnh để chạy `Graphonomy`, ta được file python thực thi. Khi chạy toàn bộ phương pháp, ta chỉ cần kích hoạt môi trường thực thi, chỉ dẫn thư mục thực thi đến thư mục Repository `Graphonomy` và chạy file `run.py` (Miniconda : `python run.py`)

   Sau đây là đoạn code `run.py`:

```
from PIL import Image
import os

def resize_image(input_path, output_path, scale_percent):
    # Read the original image
    image = Image.open(input_path)

    # Calculate new size based on ratio
```

```
    width = int(image.width * scale_percent / 100)
    height = int(image.height * scale_percent / 100)
    new_size = (width, height)

    # Resize image
    resized_image = image.resize(new_size)

    # Save new image
    resized_image.save(output_path)

# Example

def main():
    if os.path.exists("img/resize") == False: #  t o  folder resize
        os.mkdir("img/resize")
    if os.path.exists("img/image-parse") == False: #  t o  folder parse
        os.mkdir("img/image-parse")

    #Resize image and parsing
    for filename in os.listdir("img/model"):

        resize_image("img/model/"+filename, "img/resize/" + filename, 50)
        input_path = "./img/resize/" + filename
        output_path = "./" + filename[:-4]

  #Run human parsing
        os.system("python exp/inference/inference.py --loadmodel ./
    inference.pth --img_path " + input_path + " --output_path ./img/image-
    parse --output_name " + output_path)

        # Resize parsed-image to original size
        resize_image("img/image-parse/"+filename[:-4]+".png", "img/image-
    parse/" + filename[:-4]+".png", 200)

        # Delete unnecessary gray files
        os.remove("img/image-parse/" + filename[:-4] + "_gray.png")

main()
```

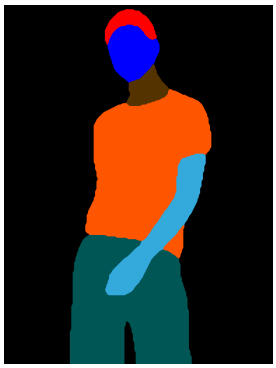
5.3.4 Demo

Link demo Human Parsing: https://youtu.be/rGicF1xxLw4

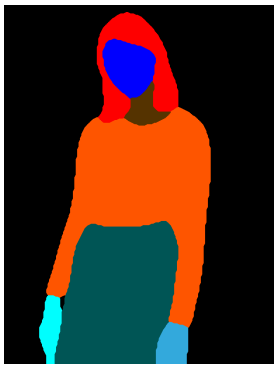
5.3.5 So sánh

Sau đây là kết quả so sánh segmentation map giữa dữ liệu của tác giả (ảnh gốc) và kết quả Human Parsing của nhóm (S-map): [1]

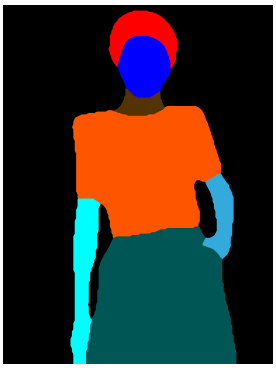
[1] Tác giả VITON-HD sử dụng phương pháp CIHP_PGN - năm 2018, còn nhóm sử dụng Graphonomy - năm 2019 nên kết quả Human Parsing được cải thiện hơn.



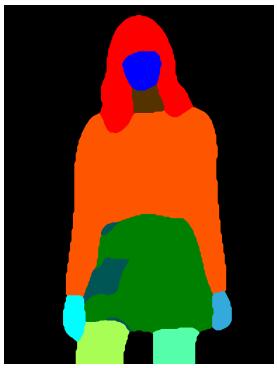
Hình 11: Ảnh gốc 1



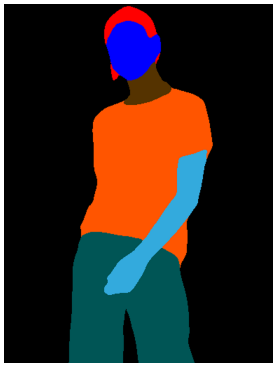
Hình 12: Ảnh gốc 2



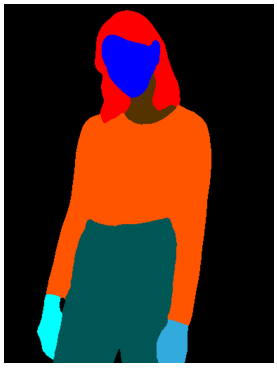
Hình 13: Ảnh gốc 3



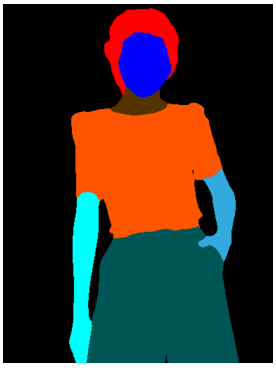
Hình 14: Ảnh gốc 4



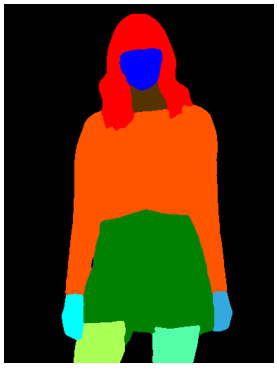
Hình 15: S-map 1



Hình 16: S-map 2



Hình 17: S-map 3

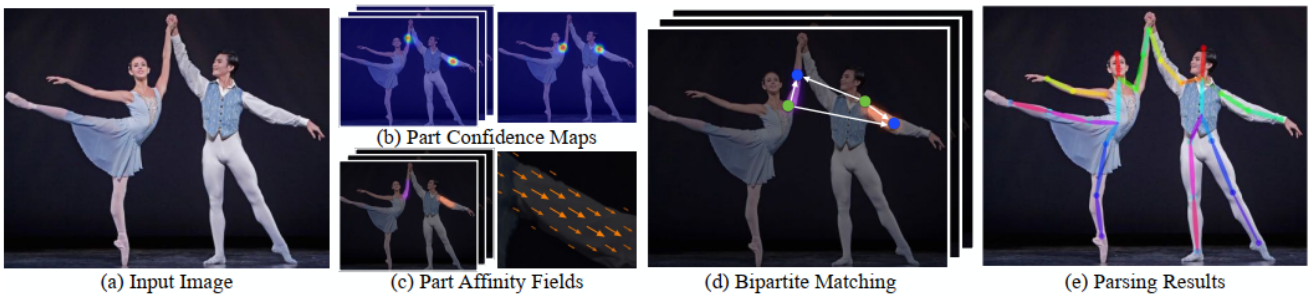


Hình 18: S-map 4

5.4 Rút trích đặc trưng dáng người (pose estimation)

Mô hình được sử dụng: OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields[9]

5.4.1 OpenPose



Hình 19: Cách hoạt động của model OpenPose: (a) Sử dụng hình ảnh làm đầu vào cho một mạng nơ-ron tích chập (CNN) để dự đoán chung (b) bản đồ độ tin cậy cho việc phát hiện các phần cơ thể (PCM) và (c) các trường liên kết phần cơ thể (PAFs) cho việc kết hợp các phần cơ thể. Sau đó (d) phân tích một tập hợp các tương quan hai phía để kết hợp các phần cơ thể. (e) Cuối cùng, tạo ra các tư thế cơ thể đầy đủ cho tất cả mọi người trong hình ảnh.

Quy trình tổng thể mô hình OpenPose: Hệ thống nhận vào một hình ảnh màu có kích thước 1024×768 (Hình a) và tạo ra các đường 2D mô phỏng dáng người cho mỗi người trong hình ảnh (Hình e).

- Đầu tiên, một mạng truyền chuyển tiếp dự đoán một tập hợp các bản đồ độ tin cậy 2D (S) cho vị trí các phần cơ thể (Hình b) và một tập hợp các trường vector 2D (L) của các trường liên kết phần cơ thể (PAFs) - mã hóa mức độ kết hợp giữa các phần (Hình c).
- Tập hợp $S = (S_1; S_2; ...; S_J)$ có J bản đồ độ tin cậy.
- Tập hợp $L = (L_1; L_2; ...; L_C)$ có C trường vector 2D.
- Cuối cùng, các bản đồ độ tin cậy (S) và các trường liên kết phần cơ thể (PAFs) (L) được phân tích bằng phương pháp suy luận tham lam (Hình d) để tạo ra các điểm chính 2D cho tất cả mọi người trong hình ảnh.

5.4.2 Input, Output

Input (dữ liệu đầu vào) và Output (dữ liệu đầu ra) của tác vụ Pose Estimation:

- Input: Ảnh người, độ phân giải 1024x768
- Output: Ảnh bản đồ dáng người (pose map `P`), độ phân giải 1024x768 và tập tin `json` gồm tập hợp các điểm chính thể hiện dáng người.

5.4.3 Cài đặt và hướng dẫn sử dụng

Vì OpenPose có bản portable nên không cần thực hiện trên môi trường thực thi đã tạo ở Tuần 1 hoặc Tuần 2.

1. Tải OpenPose bản portable :
   https://github.com/CMU-Perceptual-Computing-Lab/openpose/releases [13]
   - Tải 1 trong 2 bản : tối ưu cho GPU hoặc chỉ dùng CPU
   - Giải nén được thư mục `openpose(1)`
2. Tải các model của OpenPose:
   https://drive.google.com/uc?id=1QCSxJZpnWvM00hx49CJ2zky7PWGzpcEh [14]
   - Giải nén được thư mục `models(2)`, sao chép vào thư mục `models` nằm trong thư mục `openpose(1)`
3. Dùng Windows PowerShell để chạy theo hướng dẫn sau:
   https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/01_demo.md [15]
   - Chuyển thư mục làm việc đến thư mục `openpose` (`cd openpose`)
   - bin\OpenPoseDemo.exe --image_dir examples/test/ --hand --write_images examples/output/

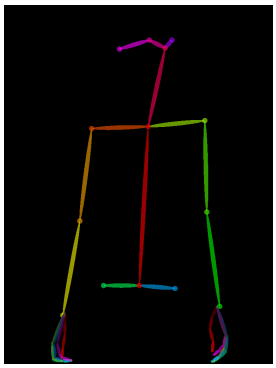
5.4.4 Demo

Link demo Pose Estimation: https://youtu.be/rGicF1xxLw4

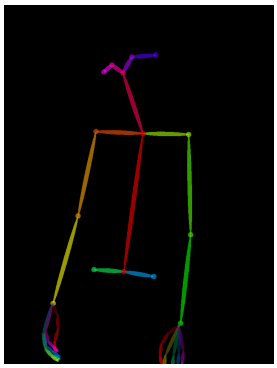
5.4.5 So sánh

Sau đây là kết quả so sánh pose map giữa dữ liệu của tác giả (ảnh gốc) và kết quả pose estimation của nhóm (P-map): [2]

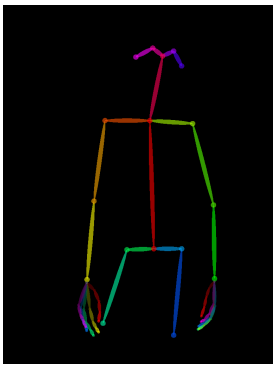
[2]Vì dùng chung 1 phương pháp OpenPose nên kết quả của tác giả VITON-HD và của nhóm khá giống nhau.



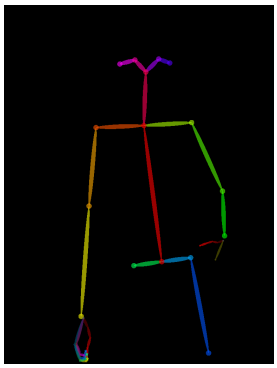
Hình 20: Ảnh gốc 1



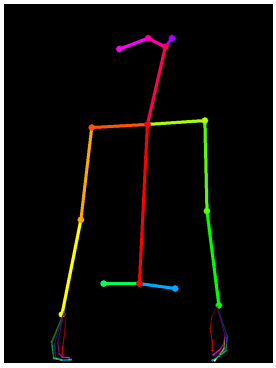
Hình 21: Ảnh gốc 2



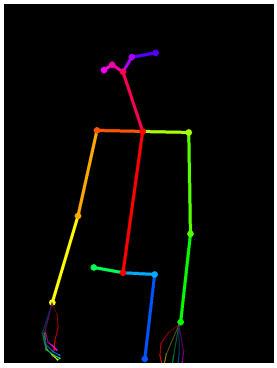
Hình 22: Ảnh gốc 3



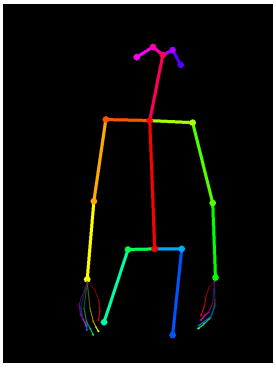
Hình 23: Ảnh gốc 4



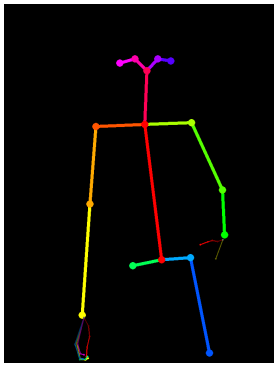
Hình 24: P-map 1



Hình 25: P-map 2



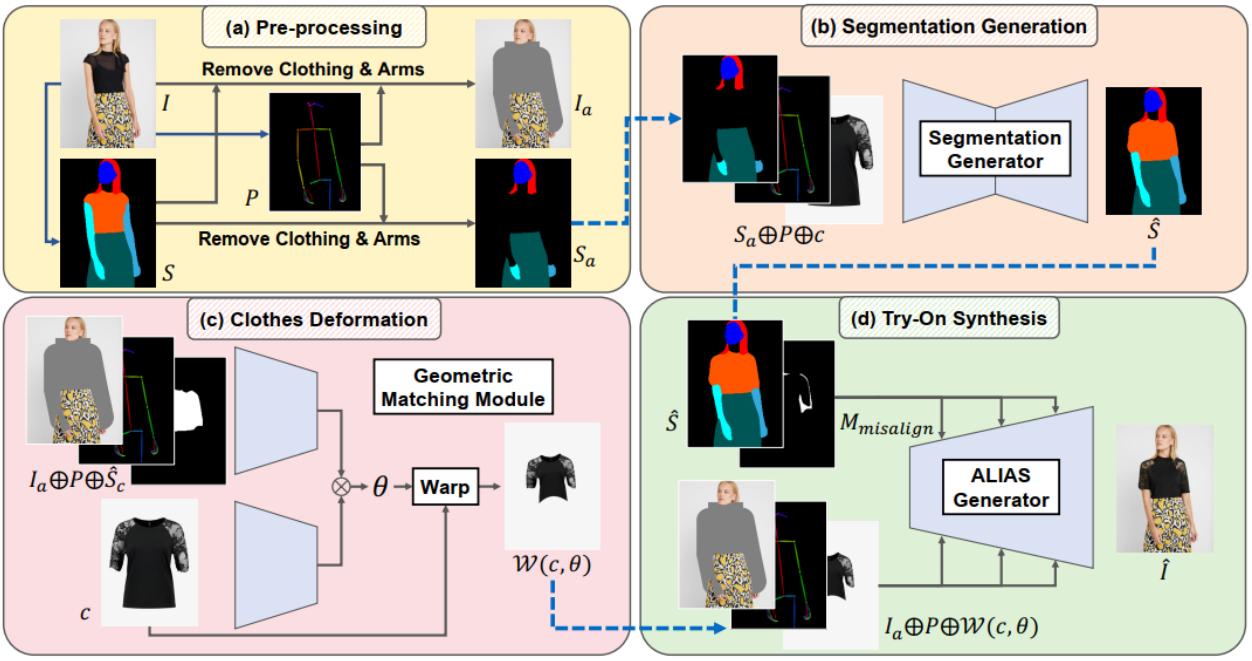
Hình 26: P-map 3



Hình 27: P-map 4

6 Tuần 3

6.1 Mô hình VITON-HD



Hình 28: Tổng quan mô hình VITON-HD

6.1.1 Tổng quan VITON-HD

VITON-HD là một phương pháp mới trong lĩnh vực thử đồ ảo dựa trên hình ảnh. VITON-HD có khả năng chuyển đổi một món đồ cụ thể lên khu vực tương ứng trên cơ thể của một người. VITON-HD tạo ra các hình ảnh thử đồ ảo có độ phân giải cao, đạt 1024×768 pixel, vượt qua giới hạn độ phân giải thấp của các phương pháp hiện có.

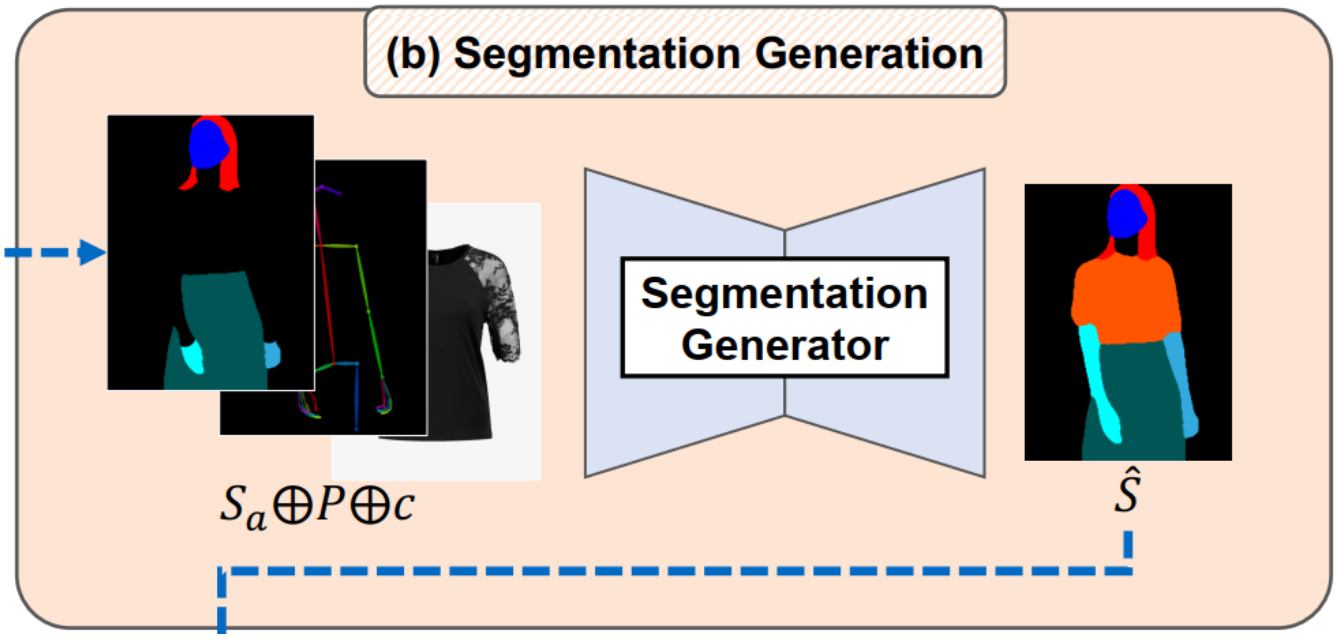
6.1.2 Input, Output

- Input: Ảnh người (`I`), bản đồ dáng người (`P` - định dạng `json` và `png`), bản đồ phân đoạn người (`S`), ảnh quần áo (`c`) và mask quần áo ($\texttt{M}_\texttt{c}$).
- Output: Ảnh người mặc quần áo mới ($\hat{I}$)

6.1.3 Cách cài đặt

Từ môi trường thực thi đã chuẩn bị ở Tuần 1 và các đầu vào, ta sẽ khởi chạy mô hình VITON-HD

6.2 Pre-trained model: Segmentation Generator



Hình 29: Segmentation Generation

6.2.1 Ý nghĩa mô hình

Mô hình Segmentation Generator được sử dụng để tạo ra bản đồ phân đoạn người (segmentation map) ($\hat{S}$) mặc đồ mới (c) . Bản đồ phân đoạn này sẽ định vị và xác định các vùng của cơ thể như khuôn mặt, tay, chân, áo quần, để hướng dẫn quá trình tạo hình áo quần sau này.

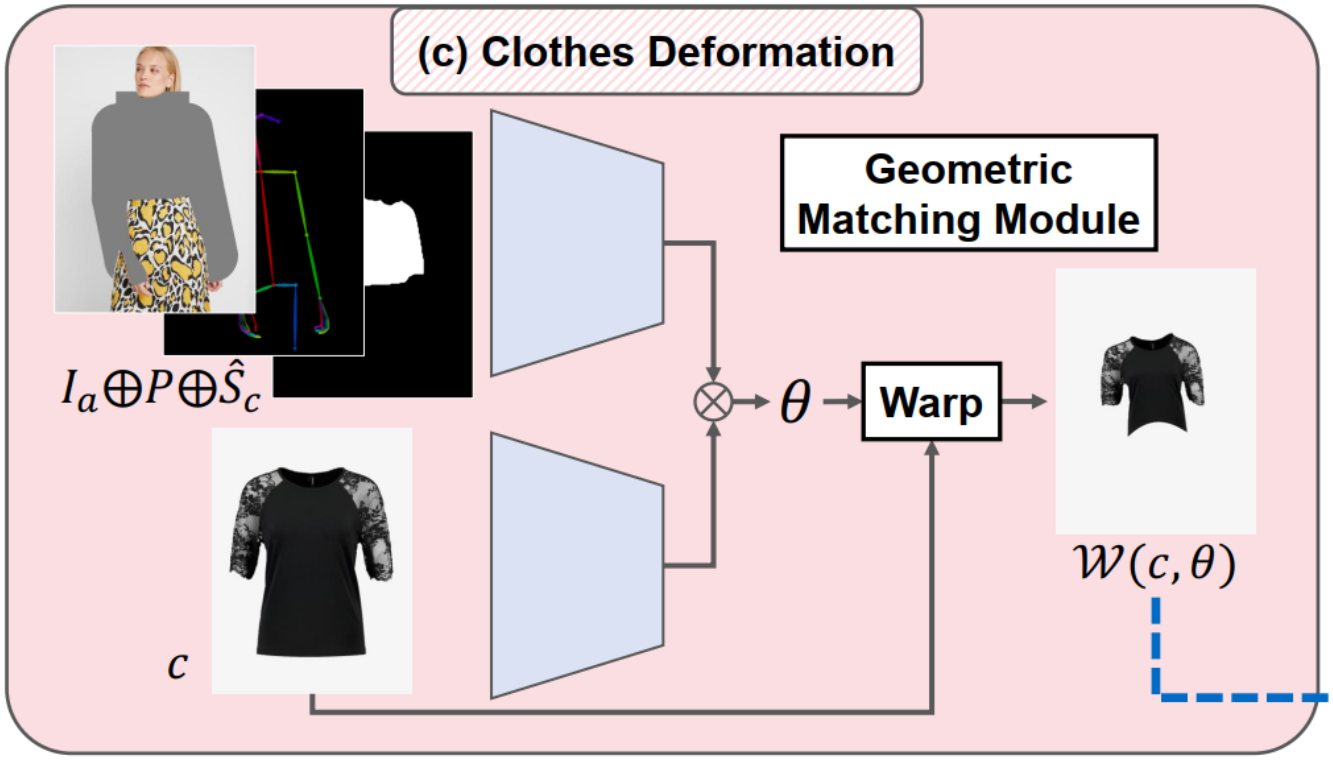
6.2.2 Quá trình huấn luyện

Mô hình Segmentation Generator (G_s) dựa trên kiến trúc U-Net [16] sẽ được huấn luyện để học một hàm ánh xạ giữa S và (S_a, P, c). Quá trình huấn luyện sẽ tối ưu hóa mô hình để dự đoán chính xác bản đồ phân đoạn người mặc đồ mới ($\hat{S}$).

6.2.3 Input và Output

- Input: Bản đồ phân đoạn người (S_a), bản đồ dáng người (P), ảnh quần áo cần thay (c)
- Output: Bản đồ phân đoạn người mặc đồ c ($\hat{S}$)

6.3 Pre-trained model: Geometric Matching Module



Hình 30: Clothes Deformation

6.3.1 Ý nghĩa mô hình

Mô hình Geometric Matching Module được sử dụng để tỉnh chỉnh ảnh quần áo `c` sao cho phù hợp với ($\hat{S}_c$) - phần áo quần của ($\hat{S}$). Quá trình này giúp đảm bảo rằng quần áo sẽ được "đúng vị trí" trên cơ thể mặc dù có sự khác biệt về pose và hình dạng giữa người trong ảnh quần áo và ảnh người.

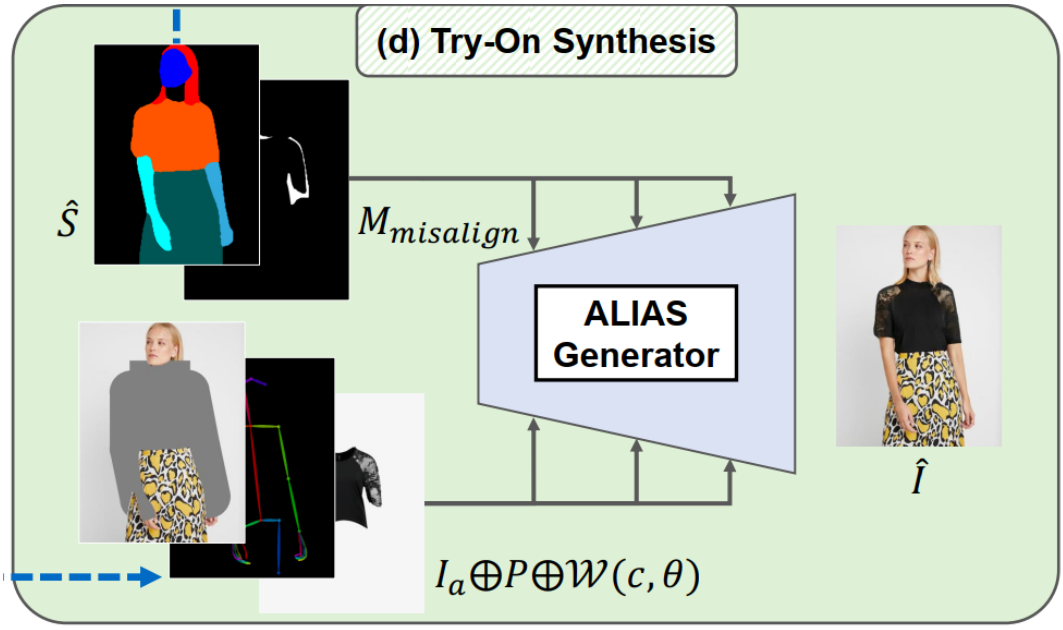
6.3.2 Quá trình huấn luyện

Mô hình Geometric Matching Module dùng model được giới thiệu trong CPVTON [17] với ($\mathtt{I_a}$, `P`) và $\hat{S}_c$ làm đầu vào. Một ma trận tương quan (correlation matrix) giữa các đặc trưng được trích xuất từ ($\mathtt{I_a}$, `P`) và `c` được tính toán đầu tiên. Với ma trận tương quan làm đầu vào, mạng nơ-ron hồi quy (regression network) dự đoán các tham số chuyển đổi TPS θ, và ảnh quần áo `c` được tinh chỉnh bởi θ. Trong quá trình huấn luyện, mô hình lấy $\mathtt{S_c}$ được trích xuất từ `S` thay vì $\hat{S}_c$. Huấn luyện mô hình sẽ tìm ra ánh xạ hợp lý để tinh chỉnh áo quần `c` sao cho phù hợp với cơ thể người.

6.3.3 Input và Output

- Input: Hình ảnh người không phụ thuộc vào quần áo ($\mathtt{I_a}$), bản đồ dáng người (`P`), vùng quần áo của $\hat{S}$ ($\hat{S}_c$), ảnh quần áo cần thay (`c`).
- Output: Ảnh quần áo được tinh chỉnh để vừa với cơ thể người ($\mathtt{W(c,}\theta\mathtt{)}$)

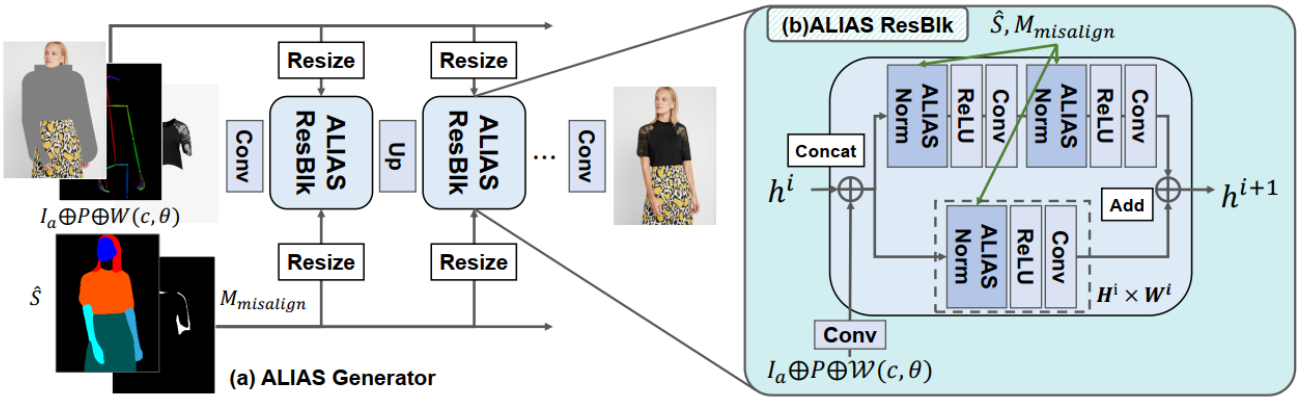
6.4 Pre-trained model: ALIAS Generator



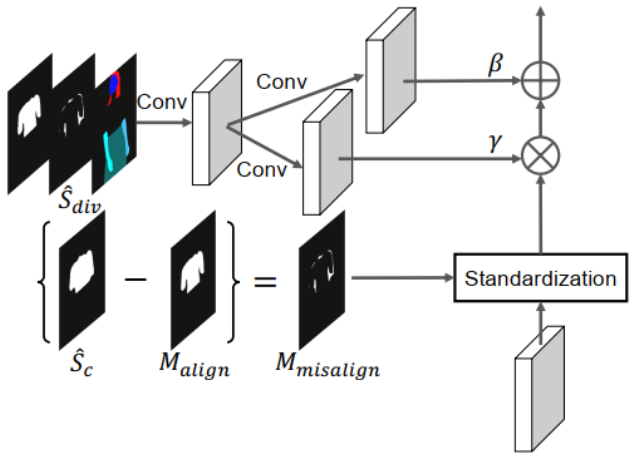
Hình 31: Try-On Synthesis

6.4.1 Ý nghĩa mô hình

Mô hình ALIAS Generator được sử dụng để tạo ra ảnh tổng hợp của người mặc đồ mới ($\hat{I}$) từ các output của các bước trước đó. Mô hình kết hợp ảnh người không phụ thuộc vào quần áo, bản đồ dáng người ($\texttt{I}_\texttt{a}$, $\texttt{P}$) với ảnh quần áo đã được tinh chỉnh ($\texttt{W(c,}\theta\texttt{)}$), theo chỉ dẫn của $\hat{S}$. ($\texttt{I}_\texttt{a}$, $\texttt{P}$, $\texttt{W(c,}\theta\texttt{)}$) được đưa vào từng lớp của ALIAS Generator. Đối với $\hat{S}$, một phương pháp chuẩn hóa có điều kiện mới gọi là ALIgnment-Aware Segment (ALIAS) normalization được sử dụng. Chuẩn hóa ALIAS cho phép bảo tồn thông tin ngữ nghĩa và loại bỏ thông tin sai lệch từ các vùng không cùng trục bằng cách tận dụng $\hat{S}$ và mask của những vùng này.



Hình 32: ALIAS generator



Hình 33: ALIAS normalization

Tóm lại, mô hình xử lý các vùng không tương ứng (misaligned) giữa áo quần và cơ thể người để tạo ra kết quả tổng hợp chất lượng cao (1024×768), bao gồm cả việc bảo tồn chi tiết của áo quần trong ảnh đầu ra.

6.4.2 Quá trình huấn luyện

Mô hình ALIAS Generator có thể được huấn luyện bằng cách sử dụng các cặp ảnh gốc và ảnh mục tiêu, cùng với bản đồ phân đoạn và ánh xạ hình học tương ứng. Quá trình huấn luyện tập trung vào việc học cách xử lý các vùng không tương ứng (misaligned) và bảo tồn chi tiết của áo quần trong quá trình tổng hợp ảnh.

6.4.3 Input và Output

- Input: Bản đồ phân đoạn người mặc đồ $\mathtt{c}$ ($\hat{S}$), mask sai lệch nhị phân ($\mathtt{M_{misalign}}$), hình ảnh người không phụ thuộc vào quần áo ($\mathtt{I_a}$), bản đồ dáng người ($\mathtt{P}$), ảnh quần áo được tinh chỉnh vừa với cơ thể người ($\mathtt{W(c,\theta)}$).

  Mask sai lệch nhị phân - misalignment binary mask ($\mathtt{M_{misalign}}$) được tạo bằng cách loại trừ mask của ảnh quần áo $\mathtt{c}$ đã được tinh chỉnh ($\mathtt{W(M_c,\theta)}$) khỏi vùng quần áo của $\hat{S}$ ($\hat{S}_c$) theo công thức sau:

  1. $\mathtt{M_{align}} = \hat{S}_c \cap \mathtt{W(M_c,\theta)}$, trong đó M_c là mask của quần áo $\mathtt{c}$ cần thay
  2. $\mathtt{M_{misalign}} = \hat{S}_c$ - $\mathtt{M_{align}}$

- Output: Ảnh người mặc đồ mới ($\hat{I}$)

6.5 Demo

Link demo VITON-HD: https://youtu.be/b7qWzQfCEQU

6.6 So sánh

Sau đây là kết quả khi chạy mô hình VITON-HD với các đầu vào mà nhóm đã chuẩn bị so sánh với dữ liệu của tác giả VITON-HD.

Hình 34: Ảnh gốc 1

Hình 35: Ảnh gốc 2

Hình 36: Ảnh gốc 3

Hình 37: Ảnh gốc 4

Hình 38: V-HD 1

Hình 39: V-HD 2

Hình 40: V-HD 3

Hình 41: V-HD 4

7 Tổng kết - Nhận xét

Tổng kết

- Sử dụng và cài đặt thành thạo các môi trường thực thi mà đồ án yêu cầu.
- Giải thích và mô tả rõ ràng các khái niệm, cách thức hoạt động của đồ án.
- Các sản phẩm về human parsing, pose estimation, virtual try-on có độ chính xác cao khi so sánh với các dữ liệu của tác giả.
- Đồ án hoàn thành đúng tiến độ và có độ hoàn thiện tốt.

Nhận xét

- Trong thời gian thực hiện đồ án, các thành viên có tinh thần trách nhiệm và năng lực tốt, hoàn thành nhiệm vụ được giao đúng với thời hạn.
- Có sự nỗ lực lớn, tinh thần đoàn kết giúp đỡ nhau giữa các thành viên.

Tài liệu

[1] Conda.
https://docs.conda.io/projects/conda/en/stable/.

[2] Miniconda.
https://docs.conda.io/projects/miniconda/en/latest/.

[3] Managing environments.
https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html.

[4] Installation VITON-HD — Official PyTorch Implementation.
https://github.com/shadow2496/VITON-HD#installation.

[5] Viton-hd: High-resolution virtual try-on via misalignment-aware normalization.
https://github.com/shadow2496/VITON-HD.

[6] Dataset.
https://www.dropbox.com/s/10bfat0kg4si1bu/zalando-hd-resized.zip?dl=0.

[7] ANGIE LEE. What is a pretrained ai model?
https://blogs.nvidia.com/blog/what-is-a-pretrained-ai-model/.

[8] Ke Gong, Yiming Gao, Xiaodan Liang, Xiaohui Shen, Meng Wang, and Liang Lin. Graphonomy: Universal human parsing via graph transfer learning. In *CVPR*, 2019.

[9] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[10] Pascal-Person-Part trained model.
https://drive.google.com/file/d/1E$_{Vg}$*VDWfAJFPfe* − *LLu2RQaYQMdhjv9h/view*.

[11] CIHP trained model.
https://drive.google.com/file/d/1eUe18HoH05p0yFUd$_s$*N6GXdTj82aW0m9/view*.

[12] CUDA Toolkit 11.8.
https://developer.nvidia.com/cuda-11-8-0-download-archive/.

[13] OpenPose bản portable.
https://github.com/CMU-Perceptual-Computing-Lab/openpose/releases.

[14] Models của OpenPose.
https://drive.google.com/uc?id=1QCSxJZpnWvM00hx49CJ2zky7PWGzpcEh.

[15] Hướng dẫn sử dụng OpenPose.
https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/01_demo.md.

[16] Philipp Fischer Olaf Ronneberger and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2015.

[17] Xiaodan Liang Yimin Chen Liang Lin Bochao Wang, Huabin Zheng and Meng Yang. Toward characteristic-preserving image-based virtual try-on network. In *ECCV*, 2018.