

POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria del Cinema e dei Mezzi di Comunicazione

Tesi di Laurea
**Creazione di avatar da immagini mediante tecniche
di intelligenza artificiale**



Relatori

prof. Andrea Sanna
prof. Federico Manuri

Candidata

Lisa Trinchieri

ANNO ACCADEMICO
2024-2025

Sommario

Questo lavoro di tesi si propone di sviluppare un metodo per la generazione di avatar 3D partendo da una singola immagine RGB acquisita in condizioni non controllate (in-the-wild), utilizzando approcci avanzati di intelligenza artificiale e modellazione tridimensionale. L'obiettivo è quello di ottenere una mesh umanoide tridimensionale il più possibile realistica, superando le limitazioni dei modelli tradizionali.

Dopo una panoramica sugli avatar e le loro applicazioni, verranno analizzati i principali metodi di ricostruzione (nello specifico, model-based e model-free, con le relative sottocategorie di cui si compongono). Seguirà un'analisi dello stato dell'arte (SOTA) dei progetti e dei dataset esistenti, suddivisi in base agli elementi fondamentali della ricostruzione (per il corpo, i vestiti, la testa e i capelli), al fine di individuare quelli ottimali per questo progetto. Dopo aver definito i requisiti, verranno descritte nello specifico le reti neurali più adatte nel nostro caso specifico e le tecnologie utilizzate nel workflow del progetto, tra cui i software, le librerie ecc.. La parte fondamentale della tesi consiste nella descrizione della pipeline sviluppata, che dettaglia le varie fasi del processo, dall'acquisizione dell'immagine fino alla generazione della mesh finale, evidenziando, per ogni step effettuato, le scelte implementative e le problematiche affrontate.

Infine, vengono analizzati i risultati ottenuti, mettendo in luce sia le prestazioni del sistema che le sue limitazioni. La tesi si conclude con una riflessione sui possibili sviluppi futuri, proponendo migliorie volte a incrementare l'accuratezza e la fedeltà della ricostruzione.

Indice

1	Introduzione agli avatar	4
1.1	Introduzione	4
1.2	Avatar	5
1.2.1	Cosa sono	6
1.2.2	Utilizzi	7
1.3	Metodi di ricostruzione	9
1.3.1	Metodi model-based	10
1.3.2	Metodi Model-free	15
2	Stato dell'Arte	19
2.1	Ricostruzione del corpo	19
2.2	Ricostruzione dei vestiti	25
2.3	Ricostruzione corpo e vestiti	31
2.4	Ricostruzione del volto	33
2.5	Ricostruzione dei capelli	38
2.6	Datasets	41
2.6.1	Dataset di corpi	41
2.6.2	Dataset di vestiti	42
2.6.3	Dataset di volti/espressioni	44
2.6.4	Dataset di acconciature	45
3	Strumenti utilizzati	47
3.1	Prerequisiti	47
3.2	Reti e dataset	48
3.2.1	Corpo	48
3.2.2	Vestiti	50
3.2.3	Viso	53
3.2.4	Capelli	55
3.2.5	Segmentazione	56
3.2.6	Dataset	57
3.3	Tecnologia utilizzata	59
3.3.1	Sistema Operativo	59
3.3.2	Python	59

3.3.3	Anaconda	61
3.3.4	Pycharm	62
3.3.5	Blender	63
3.3.6	AI Background Remover	64
4	Pipeline di lavoro	65
4.1	Pipeline	65
4.2	Input	67
4.3	Step Zero	68
4.3.1	Crop.py	68
4.3.2	Remove.py	69
4.3.3	Move_images.py	70
4.3.4	Run_networks.py	70
4.3.5	Move_objs.py	71
4.3.6	Run_project.py	71
4.4	Step Uno	71
4.4.1	Step 1.1	71
4.4.2	Step 1.2	74
4.4.3	Step 1.3	78
4.5	Step Due	79
4.6	Step Tre	85
4.7	Step Finale	92
5	Risultati ottenuti	94
5.1	Output	94
5.2	Metriche di valutazione	94
5.2.1	Misura di distorsione percettiva basata sulle caratteristiche spaziali	95
5.2.2	Misura di distorsione percettiva spaziale ponderata dalla velocità	96
5.2.3	Integrazione delle caratteristiche temporali e della metrica DMPD	97
5.2.4	Analisi delle metriche	99
5.2.5	Metriche qualitative	101
5.3	Limiti del modello	102
6	Conclusioni e sviluppi futuri	104
6.1	Conclusioni	104
6.2	Sviluppi futuri	105

Capitolo 1

Introduzione agli avatar

1.1 Introduzione

La modellazione 3D svolge un ruolo essenziale in contesti quali la computer vision e la computer grafica, trovando applicazione in numerosi ambiti di ricerca e dell'industria. I recenti progressi nelle rappresentazioni neurali e nei modelli generativi hanno contribuito ad accelerare lo sviluppo di nuove tecniche, favorendo significativi avanzamenti nella ricostruzione tridimensionale. In particolare, negli ultimi anni si è assistito ad un incremento dell'interesse nei confronti della modellazione di mesh umanoidi tridimensionali, con l'obiettivo di perfezionare le metodologie esistenti e introdurre soluzioni sempre più performanti e accurate per la creazione di avatar digitali. Le ricerche svolte in questo contesto hanno portato ad un costante miglioramento delle tecniche di acquisizione, ricostruzione e animazione dei modelli, che ricoprono un ruolo sempre più significativo in molteplici ambiti, dal gaming all'animazione, fino alla realtà virtuale e aumentata.

Il lavoro di questa tesi si inserisce all'interno di questo contesto di ricerca in evoluzione, prestando particolare attenzione alla generazione di avatar 3D il più possibile realistici. L'obiettivo principale del progetto è, quindi, quello di sviluppare un metodo efficace per ottenere una mesh umanoide tridimensionale a partire da una singola immagine RGB scattata in condizioni non controllate (in-the-wild).

Nella prima parte dell'elaborato viene presentata una panoramica sugli avatar, sia bidimensionali che tridimensionali, in cui ne vengono esplorate le principali applicazioni nei diversi ambiti di utilizzo. Vengono quindi illustrate le principali tecniche per la ricostruzione di modelli 3D umanoidi, suddividendo i metodi in due macro-categorie: i metodi *model-based*, che si basano su modelli parametrici espliciti, come SMPL, per adattare una rappresentazione tridimensionale ad un'immagine in input; e i metodi *model-free*, che sfruttano rappresentazioni implicite per gestire dettagli più complessi, come le deformazioni dei vestiti o la variabilità delle capigliature, ga-

rantando una maggiore fedeltà visiva. Le tipologie appena descritte si suddividono a loro volta in altre sottocategorie.

Successivamente, si scenderà più nel dettaglio, fornendo un'analisi approfondita dello Stato dell'Arte (SOTA) e suddividendo la ricerca in base ai diversi aspetti della ricostruzione: il corpo, i vestiti, la testa e i capelli. Per ogni categoria vengono analizzati i metodi esistenti e i dataset disponibili, utili per l'addestramento e la valutazione dei modelli creati in output dalle varie reti. Questa fase consente di identificare i principali limiti e le potenzialità delle soluzioni esistenti, fornendo le basi per le scelte metodologiche del progetto.

Dopo aver delineato il contesto teorico, vengono definiti i requisiti specifici del progetto, a cui segue la descrizione delle reti neurali selezionate tra quelle analizzate nel SOTA, che soddisfano le esigenze progettuali. In questa sezione vengono inoltre definiti gli strumenti e le tecnologie utilizzate, come Blender per la visualizzazione e la manipolazione delle mesh, PyTorch per l'implementazione delle reti neurali, e il sistema operativo scelto per la fase di sviluppo ed esecuzione.

Il fulcro dell'elaborato è rappresentato dalla descrizione dettagliata della pipeline sviluppata, in cui vengono illustrati i vari passaggi del processo di ricostruzione, dall'acquisizione dell'immagine fino alla generazione della mesh finale. Per ogni step vengono evidenziate le scelte implementative e le problematiche affrontate.

I risultati in output vengono successivamente analizzati, al fine di mettere in luce i punti di forza e le limitazioni del progetto, come eventuali errori nella ricostruzione di dettagli particolarmente complessi o difficoltà legate alle caratteristiche delle immagini in input.

Infine, la tesi si conclude con una riflessione sui possibili sviluppi futuri, proponendo implementazioni e miglioramenti per ottenere una ricostruzione ancora più accurata e realistica.

Nello specifico, questo elaborato è strutturato in sei capitoli, ognuno dei quali si occupa di uno dei macro argomenti appena descritti.

1.2 Avatar

L'interesse verso la modellazione di avatar tridimensionali è cresciuto parallelamente all'evoluzione delle tecnologie digitali, spingendo la ricerca verso soluzioni sempre più sofisticate. Questo ha portato allo sviluppo di nuovi approcci per la rappresentazione e l'animazione di figure umanoidi, con l'obiettivo di migliorare il realismo e l'adattabilità dei modelli generati. L'impiego di tali tecniche si estende a diversi settori, dalla realtà virtuale alla produzione cinematografica, fino ad arrivare alle

applicazioni interattive, dove la qualità e la fedeltà della ricostruzione rivestono un ruolo chiave.

1.2.1 Cosa sono

Gli avatar sono delle rappresentazioni grafiche di un utente, del suo personaggio o della sua personae.

Per quanto riguarda i formati degli avatar, essi possono essere:

- **bidimensionali**, utilizzati soprattutto come icone nei forum, nelle comunità online e nei social (conosciute anche come foto profilo o *userpic*). Con l'avvento dei social media, e soprattutto di Facebook, queste userpic sono di solito foto degli utenti nella vita reale.



Figura 1.1: Esempi di avatar bidimensionali.

- **tridimensionali**, come personaggi utilizzati comunemente in videogiochi e mondi online. In questi casi, gli avatar possono non essere realistici e aderenti alla fisicità dell'utente bensì essere personaggi immaginari che ricalcano la personalità del giocatore e lo screen name da egli utilizzato.



Figura 1.2: Esempi di avatar tridimensionali.

Vi sono inoltre diversi livelli di rappresentazione di un avatar: *parziale*, in cui viene mostrata solo la parte superiore del corpo dell'utente (utile, per esempio, in ambienti di Realtà Virtuale, in cui solo le mani e il volto vengono trackati) e *full-body*, in cui i piedi della mesh vengono modellati e appoggiati sul pavimento.

Ricerche accademiche dimostrano come gli avatar influenzino la comunicazione e l'identità digitale di chi li utilizza: gli utenti, infatti, possono scegliere avatar con caratteristiche fittizie per migliorare l'interazione sociale o per ottenere l'accettazione dagli altri membri della community. Tuttavia, gli studi dimostrano come gli utenti tendano comunque ad utilizzare modelli che somigliano a loro stessi nella vita reale.

1.2.2 Utilizzi

Gli ambiti di utilizzo degli avatar, sia bidimensionali che tridimensionali sono molteplici, dall'interazione sociale online alla rappresentazione nei giochi e alla realtà virtuale. Di seguito, un breve elenco dei principali.

1. **Forum su Internet** (immagini 2D da 80x80 a 100x100 pixel)
Gli avatar rappresentano gli utenti, personalizzano i loro attributi e possono rifletterne la personalità, gli interessi o lo status sociale all'interno della community. Aiutano a identificare rapidamente l'autore del post senza leggere il nome utente.
2. **Chat e messaggistica** (immagini 2D o GIF)
Servono a rappresentare gli utenti durante le conversazioni online.
3. **Social media**
Gli avatar sostituiscono spesso le foto reali nei profili: possono essere ritratti di persone, loghi di organizzazioni o immagini rappresentative (come copertine di libri, locandine di film ecc.).
4. **Assistenti online**
Utilizzati come assistenti virtuali che forniscono servizi clienti su siti web, spesso alimentati da tecnologie di *elaborazione del linguaggio naturale* (NLP) e noti come "bot".
5. **Videogiochi**
Gli avatar rappresentano i giocatori nel mondo di gioco. In alcuni casi il personaggio è fisso mentre in altri gli utenti possono personalizzarne le caratteristiche fisiche (per esempio i *MMOG* (*Massively Multiplayer Online Games*) offrono avatar altamente personalizzabili e console come Nintendo Wii, 3DS e Switch permettono la creazione di avatar chiamati "Miis", utilizzabili in vari giochi).
6. **Mondi online non legati al gaming** (2D o 3D)
Hanno come obiettivo migliorare la comunicazione online e permettere agli utenti di sviluppare il proprio spazio virtuale senza un obiettivo predefinito, facilitando l'interazione con gli altri utenti e l'esplorazione di universi virtuali.

7. Virtual try-on

Permette di provare abiti virtualmente, reindirizzandoli su un corpo specifico, fruibili sia in 2D che in 3D. Nel primo caso vengono utilizzate reti neurali per generare immagini di persone vestite in pose diverse (attraverso segmentazioni bidimensionali o *landmarks*); nel secondo invece vengono ricostruiti abiti tridimensionali e adattati a modelli umani 3D (con particolare attenzione alle collisioni tra le due mesh, alla rappresentazione delle textures e della topologia degli abiti).



Figura 1.3: Esempio di applicazione di Virtual try-on.

8. Motion Capture

Vengono utilizzati per video chat e chiamate; servizi come Skype (attraverso plugin esterni) permettono l'uso di avatar parlanti al posto delle immagini della webcam. Tramite la *motion capture facciale* e la webcam gli avatar possono replicare i movimenti e le espressioni dell'utente.

9. AR/VR

Consentono agli utenti di rappresentare sé stessi in spazi 3D immersivi, come meeting virtuali, ambienti di gioco o showroom digitali. In AR possono essere utilizzati per interagire con elementi digitali sovrapposti al mondo reale, come provare abiti virtuali (virtual try-on) o visualizzare modifiche a spazi fisici in tempo reale.



Figura 1.4: Esempi di avatar full body per applicazioni computer-based.

1.3 Metodi di ricostruzione

Negli ultimi anni, la modellazione di avatar umani ha registrato progressi significativi, e con utilizzi che spaziano dalla **CG** (Computer Graphics) alla **VR** (Realtà virtuale) ad applicazioni mediche e scientifiche.

I primi approcci si basavano su hardware costosi e su laboriosi processi di calibrazione per produrre modelli realistici e di alta qualità; i recenti progressi hanno invece permesso di generare rapidamente avatar umani a partire da immagini, video o prompt di testo come input.

I metodi di ricostruzione degli avatar tridimensionali possono essere classificati in due macrocategorie:

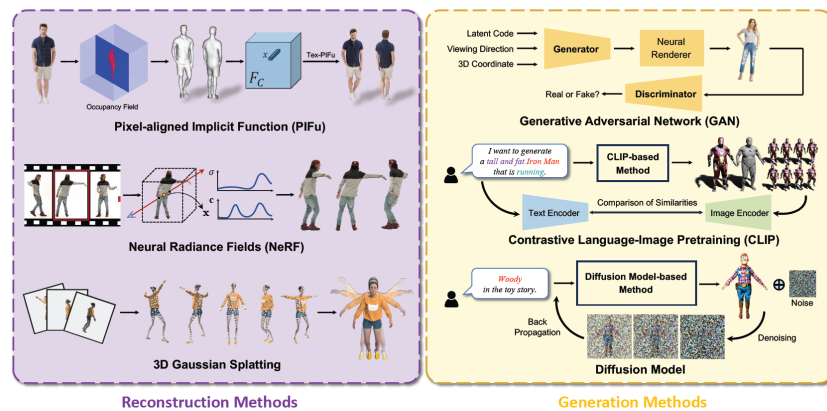


Figura 1.5: Suddivisione dei metodi di ricostruzione 3D.

- **Metodi model-based**

Utilizzano modelli parametrici espliciti (come SMPL) per adattare un'immagine in input ad un modello umano predefinito. Presentano tuttavia difficoltà nel catturare dettagli complessi, come abbigliamento e capelli.

- **Metodi model-free**

Superano le limitazioni dei metodi sopracitati e permettono di rappresentare dettagli complessi senza la necessità di adattarsi ad un modello predefinito, ma prevedendo valori di occupazione di uno spazio parametrico.

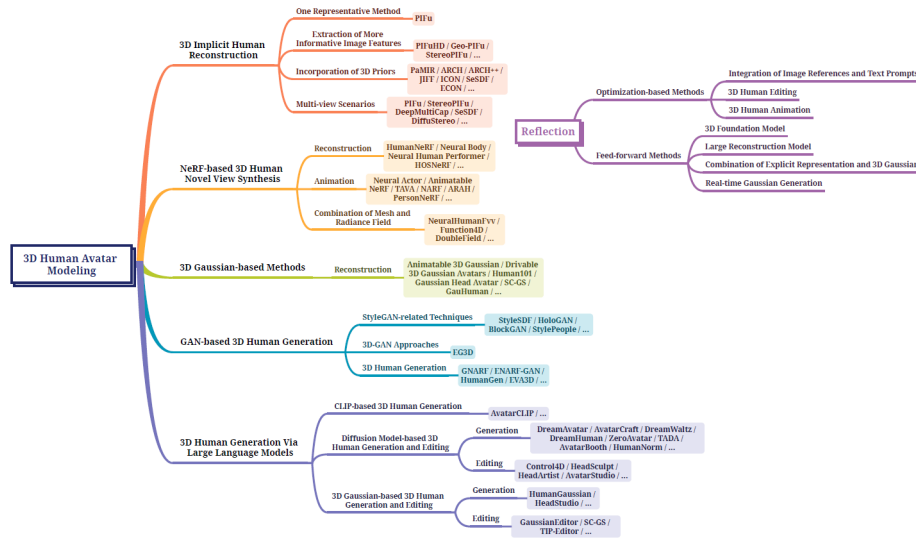


Figura 1.6: Tassonomia dei metodi di ricostruzione 3D.

1.3.1 Metodi model-based

1.3.1.1 3D Implicit Human Reconstruction

Il metodo più rappresentativo per questa categoria è **PIFu (Pixel-aligned Implicit Function)**, che fa uso di una *Multi-Layer Perception (MLP)* per modellare una funzione implicita in grado di prevedere di valore di occupazione di un punto nello spazio tridimensionale sfruttando caratteristiche pixel-aligned estratte dall'immagine di input.

Il processo prevede che, data una singola immagine o più immagini in input, il modello applichi un codificatore a clessidra e un'interpolazione bilineare per estrarre le caratteristiche pixel aligned per ogni punto 3D. Una funzione implicita predice successivamente il campo di occupazione continuo nello spazio tridimensionale. Dopo aver ottenuto la geometria sotto forma di mesh, si può utilizzare PIFU per estrarre le textures, prevedendo il valore RGB di ogni singolo vertice del modello.

Il metodo, tuttavia, presenta alcune problematiche: la mesh ricostruita infatti presenta dei dettagli geometrici sub-ottimali e di bassa qualità per essere utilizzata nelle applicazioni e risente della mancanza di informazioni derivanti da più punti di vista

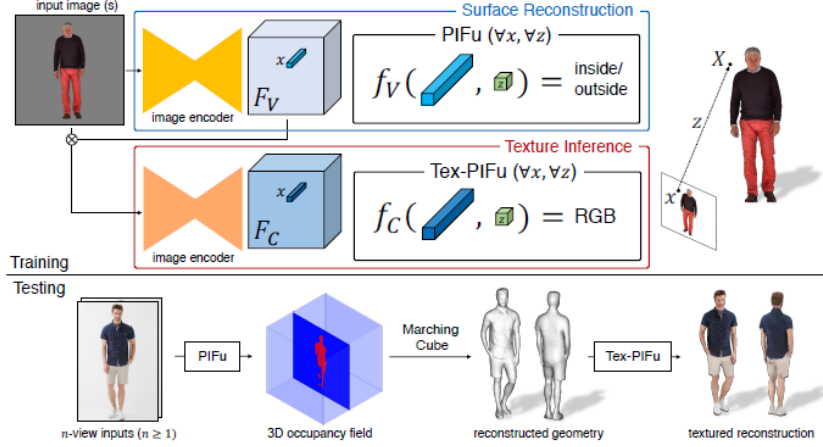


Figura 1.7: Ricostruzione 3D con PIFu e Tex-PIFu a partire da un'immagine RGB.

per quanto riguarda la modellazione di pose complesse, ambiguità nella profondità e auto occlusioni.

Per ovviare a questi problemi sono state proposte diverse soluzioni, prima tra tutte l'estrazione di un maggior numero di informazioni dall'immagine in input (a tal proposito sarebbe opportuno un miglioramento dell'encoder). Anche l'utilizzo di più immagini multi-view sarebbe efficace per una ricostruzione migliore, soprattutto per quanto riguarda i dettagli più fini dei vestiti e l'accuratezza nel catturare i movimenti. Infine, sarebbe utile incorporare caratteristiche tridimensionali in input (in particolare con SMPL e SMPLX) per migliorare la robustezza dell'output.

Nonostante le sue potenzialità, dunque, PIFu necessita di miglioramenti nella gestione delle occlusioni, nell'integrazione di metodi multi-view e nella qualità dei dataset di addestramento (scarsi e difficili da ottenere), a cui viene attribuito il peso maggiore per le problematiche riscontrate.

1.3.1.2 NeRF-based 3D Human Novel View Synthesis

NeRF è un metodo utilizzato per la sintesi di nuove viste di scene complesse. L'algoritmo presenta una scena ricostruita attraverso una rete neurale completamente connessa (non convoluzionale). A quest'ultima viene data in input una singola coordinata continua 5D (estratta da un set di immagini di cui sono note le pose della telecamera), che si compone della posizione spaziale in tre dimensioni e della direzione visuale in 2D dal punto definito dalla coordinata tridimensionale. L'output della rete è la densità volumetrica e la radianza emessa dipendente dalla vista per la specifica posizione spaziale 5D, che permettono di sintetizzare nuove viste. Utilizzando inoltre tecniche classiche di rendering volumetrico, i colori e le densità

vengono proiettati per generare un'immagine.

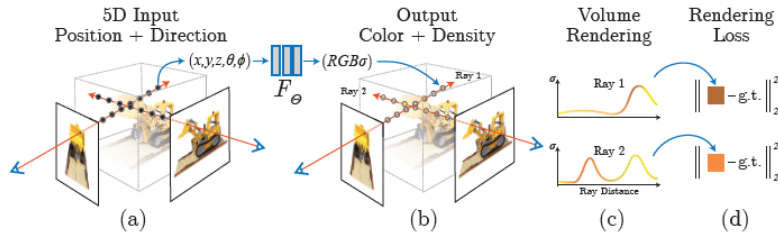


Figura 1.8: Schema del rendering volumetrico basato su input 5D, con trasformazione in colore e densità e ottimizzazione tramite rendering loss.

A differenza dei metodi di ricostruzione implicita, che necessitano di un gran numero di immagini di alta qualità per il training, NeRF riesce a raggiungere risultati foto-realistici pur utilizzando un limitato numero di immagini come dataset. Dopo aver ottenuto la densità e il colore RGB di un determinato punto nello spazio tridimensionale, la rete utilizza un modulo di rendering volumetrico per creare una scena 3D a partire da un'immagine 2D. Per ogni pixel, il rendering consiste nella definizione di un raggio r dalla posizione del pixel nella scena 3D e il campionamento dei punti tridimensionali che si trovano lungo tale raggio.

A partire da NeRF, sono stati proposti numerosi metodi di ricostruzione del corpo umano rappresentato come un campo neurale di radianza, generato senza modelli pre-addestrati o conoscenze pregresse delle reti. Oltre alla modellazione, sono state fatte ricerche anche nell'ambito dell'animazione (attraverso sequenze di pose guidate dall'utente) e sui possibili metodi per integrare sia i campi di superficie che quelli di radianza per ottenere sintesi di viste della mesh umana ad alta fedeltà.

Tra questi, alcuni metodi che sono stati sviluppati prevedono:

- **ricostruzione della mesh umanoide da telecamere statiche**

In questo scenario, tutte le telecamere utilizzate sono fisse in punti specifici e noti per catturare immagini o video del corpo target. A differenza di PIFu, che elabora soggetti umani in una sola posa, questo modello mira ad ottimizzare la scana tridimensionale, ricevendo come input più immagini con varie pose e da più punti di vista.

- **ricostruzione della mesh umanoide da telecamere dinamiche**

In aggiunta allo scenario statico appena descritto, sono stati proposti modelli che utilizzano telecamere in movimento durante il processo di acquisizione delle immagini e dei video, rendendo il modello più adatto per le applicazioni nel mondo reale.

- **animazione dei modelli 3d**

L'animazione delle mesh umanoidi si concentra soprattutto sull'apprendimento

di come varia la topologia dei vestiti in base alla posa assunta dal soggetto, e si basa su immagini multi-vista e multi-pose. Questo consente la sintesi di animazioni da punti di vista “liberi”, con sequenze di pose decise dall’utente. Tali pose possono anche non essere presenti nel dataset di allenamento del modello ma derivare direttamente da video multi-view passati come input alla rete.

- **combinazione di mesh e campo di radianza**

Basandosi su PIFu e NeRF, sono stati proposti diversi metodi per combinare i vantaggi delle mesh e dei campi di radianza per ottenere una ricostruzione dei modelli tridimensionali ad alta fedeltà.

Nonostante gli output dai metodi che utilizzano i campi di neurali siano fedeli all’originale e di alta qualità, tali risultati richiedono reti neurali costose da addestrare e onerose per quanto riguarda la fase di rendering.

1.3.1.3 3D Gaussian-based Methods

I metodi basati su 3D Gaussian Splatting (3DGS) permettono di rappresentare e renderizzare scene complesse ricevute in input in tempi piuttosto brevi, senza tuttavia compromettere la qualità dell’output o la velocità della rete. Seguendo i principi base della 3DGS, numerosi metodi hanno migliorato la ricostruzione di avatar umani nello spazio tridimensionale, producendo modelli animabili e realistici.

A partire dagli stessi input dei metodi NeRF precedentemente descritti (tra cui,

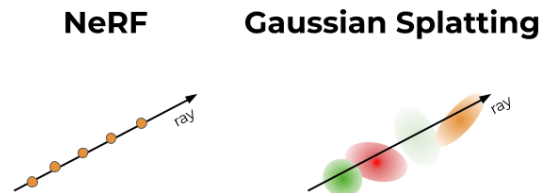


Figura 1.9: Confronto tra NeRF e Gaussian Splatting per la sintesi 3D.

i parametri della telecamera calibrati utilizzando *Structure-from-motion (SfM)*), si inizializza un set di Gaussiane 3D con le nuvole di punti create come parte del processo SfM. A differenza però di questi primi (NeRF) che utilizzano le reti neurali per sintetizzare nuove viste, i modelli che utilizzano 3DGS introducono un nuovo approccio che si basa sull’ottimizzazione delle proprietà delle Gaussiane 3D, che includono la posizione nello spazio tridimensionale, l’opacità, la covarianza anisotropa e i coefficienti armonici sferici. Tale ottimizzazione è integrata con dei passaggi di controllo adattivo della densità, durante i quali vengono aggiunti o rimossi gaussiane a seconda delle necessità, al fine di migliorare la rappresentazione della scena. Inoltre, i metodi 3DGS permettono il rendering real-time degli output che utilizza

algoritmi veloci di ordinamento sulle GPU, permettendo di eseguire un “anisotropic splatting” che rispetta l’ordine di visibilità (sfruttando l’ordinamento e il parametro di opacità estratto precedentemente dalle Gaussiane 3D) e di abilitare un backward pass rapido e accurato.



Figura 1.10: Pipeline di Gaussian Splatting: da point cloud a rendering.

Tale metodologia permette quindi di rappresentare e renderizzare scene complesse ad alta risoluzione in modo efficiente, riducendo significativamente i tempi di addestramento e di rendering. Il processo è composto, nello specifico, dai seguenti step:

- **Definizione**
Viene utilizzato, come input, un set di immagini statiche della scena e i parametri corrispondenti della telecamera.
- **Rendering**
La scena composta dalle Gaussiane 3D viene renderizzata nello spazio dell’immagine 2D, utilizzando la rasterizzazione splatting.
- **Ottimizzazione**
Per questo step vengono utilizzate tecniche *SGD* (*Stochastic Gradient Descent*). Nello specifico, viene calcolata la matrice di covarianza iniziale come un gaussiano isotropico, i cui assi sono pari alla distanza media dei tre punti vicini. Si utilizza poi un metodo di decadimento esponenziale per l’ottimizzazione.
- **Diversificazione e culling**
Per garantire dettagli sufficienti e una ricostruzione accurata della scena, 3DGS incorpora la diversificazione, per cui ogni 100 interazioni vengono aggiunti delle Gaussiane per migliorare l’output, e il culling (le Gaussiane essenzialmente trasparenti vengono rimosse nella fase di ottimizzazione).

E’ dunque evidente come i metodi basati su NeRF e su 3DGS rappresentino differenze sostanziali. Tra queste, i primi utilizzano la *MLP* (*Multi-Layer Perceptrons*) per prevedere i valori di densità e colore, mentre per le seconde vengono utilizzate direttamente le proprietà delle Gaussiane 3D. Inoltre i secondi offrono un vantaggio significativo nella fase di rendering poiché utilizzano la rasterizzazione basata su tiles per rappresentare direttamente la scena 3D. Infine, a causa delle differenze nelle

prestazioni 3D, i metodi del primo tipo generano punti non correlati, aumentando notevolmente il numero di punti e i requisiti computazionali.

1.3.2 Metodi Model-free

1.3.2.1 GAN-based 3D Human Generation

Lo sviluppo delle *Generative Adversarial Networks* (GAN) ha introdotto un nuovo processo per la generazione di avatar umanoidi, nonché nuovi metodi che si basano proprio su questa tipologia di reti, producendo mesh 3D a partire da immagini 2D.

Nello specifico, una rete generativa avversaria (GAN) si compone di due parti principali: il generatore e il discriminatore.

Il generatore riceve in input il rumore casuale e lo converte in campioni di dati complessi, come testo o immagini. I dati del training della rete vengono utilizzati per allenare diversi livelli di parametri, indispensabili per la creazione degli output. Come risultato, il generatore cerca di produrre campioni che imitano fedelmente i dati reali. Durante tutto il processo, tale componente viene addestrato utilizzando la backpropagation per regolare e rifinire i suoi parametri. Ciò che rende un generatore efficiente è la sua capacità di generare campioni sintetici di alta qualità e di diversa natura, che siano in grado di ingannare il discriminatore.

L'obiettivo di quest'ultimo è quello di distinguere tra input creati dal generatore e input reali. Valutando i campioni ricevuti in ingresso alla rete e assegnando a questi la probabilità di autenticità, il discriminatore funziona come un classificatore binario. Nel tempo, questo componente impara a distinguere tra dati autentici reali e campioni artificiali. La continua interazione tra generatore e discriminatore consente a quest'ultimo di affinare progressivamente i suoi parametri e aumentare il suo livello di competenza, permettendo inoltre alla GAN di produrre via via dati sintetici dall'aspetto sempre più realistico e complesso.

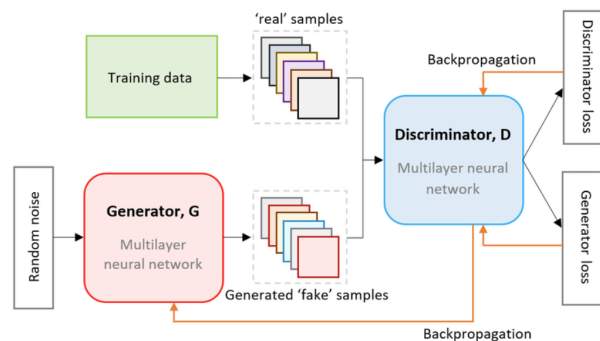


Figura 1.11: Architettura di una GAN.

I metodi che si basano sulle GAN si compongono di due componenti chiave:

- **L'architettura StyleGAN** che estende il modello tradizionale delle GAN introducendo uno spazio “style” per controllare gli attributi delle immagini generate. Questo componente è costituito da: una rete di mapping che, dato un codice latente in input, predice un segnale di controllo, e un generatore, che utilizza questo segnale per generare l'immagine finale tramite un modello sequenziale in più livelli che aumentano progressivamente la risoluzione. In particolare, il generatore parte da un input costante e adatta lo stile dell'immagine ad ogni livello di convoluzione, basandosi sul codice latente e controllando direttamente la consistenza delle caratteristiche dell'immagine nei vari step. Grazie al rumore introdotto direttamente in input, la rete è in grado di separare automaticamente attributi di alto livello (come posa e identità) e variazioni stocastiche (dettagli come lentiggini, capelli. ecc.), modellando la posizione dei giunti e consentendo operazioni di interpolazione. Il generatore utilizzato da questi modelli, a differenza di quelli tradizionali, offre una rappresentazione più lineare e meno confusionaria dei diversi fattori di variazione della rete.
- **La rappresentazione triplane (tridimensionale)**, la cui introduzione nei più recenti metodi basati su GAN permette di raggiungere migliori risultati sia per quanto riguarda le metriche 2D che 3D.

Nonostante i metodi basati su GAN abbiano raggiunto ottimi risultati nella ricostruzione di avatar umanoidi, presentano tuttavia delle problematiche: le reti, infatti, non riescono a ricostruire caratteristiche e dettagli che non sono presenti nel dataset di training, limitando quindi l'abilità del modello nel generare mesh 3D. Possono inoltre creare artefatti come sfocature, distorsioni e rumore nelle immagini generate.

1.3.2.2 3D Human Generation Via Large Language Models

Per sfruttare il potenziale dei modelli generativi 2D come, per esempio, **Contrastive Language-Image Pretraining (CLIP)** e i modelli di diffusione per generare contenuti tridimensionali, sono stati proposti svariati metodi che ottimizzano le rappresentazioni 3D (ad esempio, mesh, nuvole di punti, NeRF, Gaussiane 3D) basandosi esclusivamente su prompt testuali.

Nello specifico, CLIP si occupa di mappare immagini e testo nello stesso spazio, permettendone il confronto per estrarre le caratteristiche simili. Tra i metodi che si basano su questo modello c'è **DreamField**, uno dei primi metodi ad integrare le rappresentazioni tridimensionali con modelli di linguaggio pre-addestrati, al fine di valutare le similitudini tra gli output generati dai contenuti 3D e i prompt testuali. La pipeline di lavoro di tale metodo consiste in una fase di inizializzazione (in cui il

modello NeRF viene inizializzato su una sfera unitaria), una di rendering, in cui, durante ogni interazione vengono generate immagini RGB da direzioni della telecamera campionate casualmente, una di calcolo della perdita, durante la quale viene calcolata la similarità tra l'immagine renderizzata e il prompt testuale di CLIP, e una di ottimizzazione, in cui la perdita viene retropropagata per ottimizzare i parametri di NeRF. Tuttavia, a causa delle limitazioni di CLIP a comprendere completamente le descrizioni testuali, questi metodi non riescono a generare correttamente i dettagli fini e i movimenti complessi.

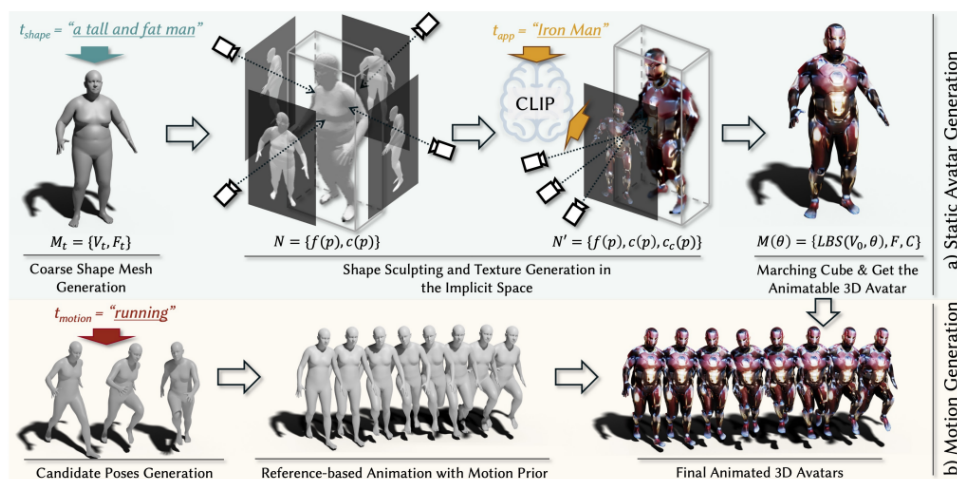


Figura 1.12: Rappresentazione della pipeline di CLIP.

Una seconda categoria di metodi sono quelli che generano mesh di umanoidi basandosi sui modelli di diffusione. Questi differiscono rispetto ai precedenti (che utilizzano CLIP e che confrontano immagini e testo mappandoli in uno spazio condiviso per comprendere e imparare le associazioni tra parole e caratteristiche dell'immagine 2D) dal momento che convertono rumore gaussiano casuale in dati strutturati attraverso un processo di Markov a cui vengono uniti una serie di step di denoising dell'output. Tuttavia, se i modelli del primo tipo possono portare ad interpretazioni imprecise di istruzioni lessicalmente complesse, quelli del secondo tipo invece, non si curano delle ambiguità linguistiche, intrinseche nelle associazioni testo-immagine, quanto piuttosto si concentrano sul perfezionamento graduale dei dati visivi, fornendo come output ricostruzioni di mesh umane più precise e dettagliate. Tuttavia, questi metodi faticano a controllare i movimenti degli umani, a ricostruire mani e piedi e creano inconsistenze nella geometria e nelle texture.

Si hanno, infine, i modelli basati sulle Gaussiane 3D per la generazione di avatar umanoidi e per il loro editing. Con lo sviluppo della 3D Gaussian Splatting, di cui si è parlato precedentemente, infatti, sono stati introdotti nuovi metodi che utilizzano la 3DGS per generare avatar tridimensionali di elevata qualità, incrementando l'ef-

ficienza del training. A differenza dei metodi di editing basati su rappresentazioni implicite come NeRF, che presentano un controllo limitato per scene complesse e una bassa velocità di ricostruzione delle stesse, i progetti che sfruttano la 3DGS offrono la possibilità di utilizzare semantiche guidate dall'utente con efficienza migliorata. Tra questi, **DreamGaussian** progetta un algoritmo di estrazione di mesh dalle Gaussiane 3D, introducendo inoltre un affinamento delle textures nello spazio UV per migliorare sia l'efficienza che la qualità dell'output.



Figura 1.13: Confronto tra le mesh crate con CLIP e con DreamField.

Capitolo 2

Stato dell'Arte

Come messo in luce nel Capitolo 1 di questo documento, gli avatar 3D vengono utilizzati per un gran numero di applicazioni: a tal proposito nel corso degli anni si sono sviluppati numerosi progetti per la ricostruzione delle mesh umanoidi. Il seguente capitolo offre una panoramica sul **SOTA, lo State Of The Art**, analizzando i metodi che sono stati creati e le loro caratteristiche.

La parte più consistente della fase di ricerca per questo progetto è stata svolta tra Febbraio e Maggio 2024, per cui le reti descritte e le considerazioni effettuate fanno riferimento allo stato dell'arte in questo arco temporale. Le ricerche sono poi state riaperte a Gennaio 2025 per miglioramenti e implementazioni per lo step di ricostruzione dei vestiti.

2.1 Ricostruzione del corpo

Per la ricostruzione del corpo umano è indispensabile tenere in considerazione due aspetti principali: la forma del corpo e la sua posa nello spazio. I metodi SOTA sono efficienti nella stima della posa, ma faticano invece a ricostruire la mesh del corpo sotto i vestiti.

Ciò avviene a causa di:

- i soli punti chiave 2D a partire da immagini o da video in input non forniscono informazioni sufficienti per definire completamente la forma del corpo 3D.
- utilizzare modelli base di shape del corpo permette di risolvere il problema al punto 1, ma limitano la ricostruzione alle sole forme base.
- è difficile per alcuni metodi definire correttamente le misure antropometriche (ad esempio stimare l'altezza delle persone rappresentate).
- la presenza di vestiti nell'immagine in input rende difficile la ricostruzione del corpo sottostante (per questo motivo viene chiesto agli utenti di indossare vestiti stretti che ben si adattino alle forme del corpo).

- i dataset con immagini in-the-wild hanno corpi 3D rumorosi e poco accurati.
- i dataset con immagini scattate in laboratorio hanno un numero ristretto di soggetti, sono limitati nelle forme che propongono e non rappresentano lo spettro completo di tutte le possibili body shapes.
- molti dataset su cui vengono allenati questi metodi si concentra su una grande varietà di pose diverse piuttosto che di forme del corpo.
- i metodi di ricostruzione parametrica come SMPL utilizzando la posizione dei giunti bidimensionali come dati di allenamento: tuttavia non è sufficiente conoscere solo la posizione dei giunti perché un gran numero di corpi diversi condivide lo stesso scheletro ma non la stessa silhouette.
- stimare i corpi da immagini 2D spesso produce corpi che si inclinano, fluttuano o penetrano il pavimento (questi metodi ignorano il fatto che i corpi sono solitamente supportati dalla scena circostante). Un motore fisico può essere utilizzato per imporre plausibilità fisica.

Per ovviare a tali problematiche sono state proposte due soluzioni principali per quanto riguarda i dati in input alla rete:

- **Creazione di immagini sintetiche**

Offrono annotazioni precise e automatiche della forma base del corpo. Questo permette un rendering più accurato delle scene tridimensionali per quanto riguarda la modellazione del corpo, la posa, i vestiti, la texture e l'illuminazione. Tuttavia, riprodurre scene realistiche con anche i capi di abbigliamento è costoso e si incontra un domain gap tra le immagini del dataset e quelle reali.

- **Cues di forma 2D da immagini in the wild** (come maschere di segmentazione, silhouettes ecc.)

Possono essere annotate manualmente o rilevate automaticamente. Tuttavia, la ricostruzione delle mesh a partire da tali informazioni tende a "gonfiare" il corpo per rappresentare anche i vestiti incorporati.

Come già anticipato, i metodi che ricostruiscono mesh 3D umanoidi, e che quindi riproducono le fattezze e il comportamento degli esseri umani in ambienti reali complessi, si dividono in due macrocategorie: **3D Human Pose Estimation (HPE)** e **Human Mesh Recovery (HMR)**.

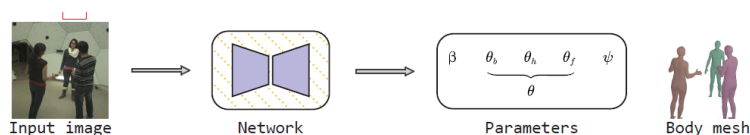
Di questi, i primi riescono a prevedere accuratamente le coordinate dei landmarks del corpo umano nello spazio tridimensionale: tale approccio permette di giungere a informazioni spaziali più complete e accurate se paragonato ai metodi bidimensionali, e permette di comprendere al meglio il comportamento umano e il suo movimento in applicazioni di Computer Vision di alto livello.

I secondi invece ricostruiscono un modello digitale del corpo in tre dimensioni, ponendo particolare attenzione a dettagli come la forma, i gesti, l'abbigliamento e le

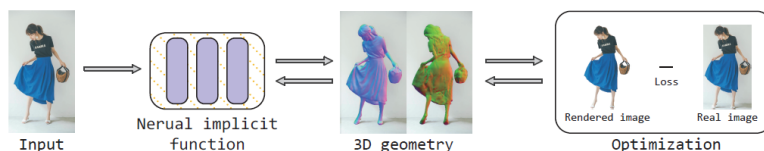
espressioni facciali, offrendo così una panoramica accurata delle interazioni tra gli esseri umani e il mondo fisico che li circonda.

Negli ultimi anni entrambe queste macrocategorie sono state oggetto di studio e di miglioramenti, permettendo ai metodi della prima tipologia di ricreare mesh e scene complesse ricevendo input di diversi tipi con più di un soggetto e a quelli della seconda di ricreare dettagli più complessi.

I metodi HMR, che ricevono in input un'immagine RGB e restituiscono in output la mesh del corpo, si dividono a loro volta in due sottocategorie: **parametrici**, che si occupano di predire i parametri di un corpo 3D statistico (SMPL, modello che verrà approfondito a breve) e **non parametrici** che prevedono una rappresentazione libera (*free-form*) del corpo umano. Di entrambi questi si è discusso diffusamente nel Capitolo 1 di questo documento.



(a) Template based human mesh recovery



(b) Template-free human mesh recovery

Figura 2.1: Differenza tra i metodi model-based e model-free.

I metodi **parametrici** si dividono a loro volta in:

- **Metodi basati sull'ottimizzazione**

Cercano le configurazioni che meglio rappresentano e si adattano all'immagine in input utilizzando marcatori 2D, come keypoints e silhouettes. Nella funzione obiettivo di tali metodi, alcuni termini penalizzano forme e pose innaturali, mentre altri, basati sui dati, misurano gli errori tra la ricostruzione della mesh e l'input 2D. Basandosi su questo paradigma, si è arrivati a vari miglioramenti del metodo introducendo nuove informazioni (tra cui giunti del corpo in 2D/3D, silhouettes, segmentazione) per evitare di incorrere nell'errore di gonfiare la mesh dei corpi rappresentare anche i vestiti, come accade alla maggior parte dei metodi. Tuttavia, nonostante i risultati ottenuti siano piuttosto allineati con l'input originale, il processo tende ad essere lento e tale metodo è fortemente influenzato dall'inizializzazione e dipendente sull'accuratezza delle pose nell'immagine 2D.

- **Metodi basati sulla regressione**

Si basano su reti neurali che vanno a regredire i parametri del modello direttamente dai pixel dell'immagine 2D ricevuta in input attraverso segnali di supervisione. Per facilitare la procedura di training del regressore, che avviene attraverso dataset di immagini da laboratorio e in-the-wild, sono state progettate diverse architetture per sfruttare rappresentazioni proxy, come silhouettes, giunti 2D e 3D, segmentazione e corrispondenze dense. Questo tipo di metodi può beneficiare dei dati sintetici e del miglioramento della stima delle rappresentazioni proxy, permettendo inoltre di sfruttare caratteristiche ad alta risoluzione.

- Infine, regressione e ottimizzazione possono essere combinati per nuovi metodi.

Una seconda categorizzazione dei metodi **HMR** divide i progetti in:

- **Linear Blend Skinning (LBS)**, anche conosciuti come **SSD (Skeletal-Subspace Deformation)**

Sono i metodi più utilizzati in cui la mesh in output viene riggata in un set di giunti che creano un albero cinematico: ogni vertice del modello tridimensionale ha un certo numero di giunti a cui viene attribuito un determinato peso (tali pesi di solito sono attribuiti manualmente da una persona fisica ma si sta cercando anche di automatizzare il processo, estraendoli dai dati a disposizione).

- **PSD (Pose space deformation)**

Ha come base LBS utilizzando in addizione delle deformazioni correttive (offset che vengono aggiunti ai vertici della mesh creata e messa in posa attraverso il metodo LBS e sono relative all'albero cinematico su cui si basa tale metodo).

Come già accennato parlando dei modelli parametrici, **SMPL** [11] è il più accurato modello 3D realistico del corpo umano, basato su tecniche di skinning e blend shapes, ed è stato sviluppato a partire da migliaia di scansioni 3D del corpo umano. Largamente utilizzato in un gran numero di progetti, sia in industria che in accademia, e testato su un gran numero di scansioni, riesce a catturare bene i parametri di posa e di forma della mesh in input, offrendo possibilità di definire il genere della mesh (M/F). Nonostante ciò, presenta delle problematiche: per far sembrare la mesh il più realistica possibile e aderente all'originale, nonostante i numerosi termini di regolarizzazione, il modello impara correlazioni spurie con i modelli presenti nel dataset di allenamento, che causano deformazioni e strani gradienti che si propagano in tutta la mesh.

Il progetto SMPL è stato recentemente migliorato in **STAR** [13] (metodo basato su LBS), che permette di avere solo un quinto dei parametri utilizzati per SMPL. Entrambi utilizzano, nella modellazione dell'output, alcuni parametri che sono dipendenti dalla forma del corpo e altri solamente dalla posa. SMPL, tuttavia, non

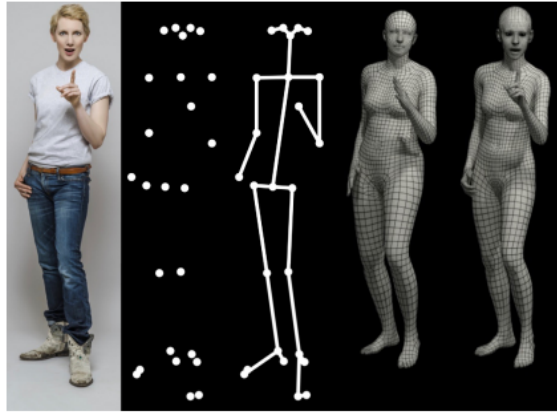


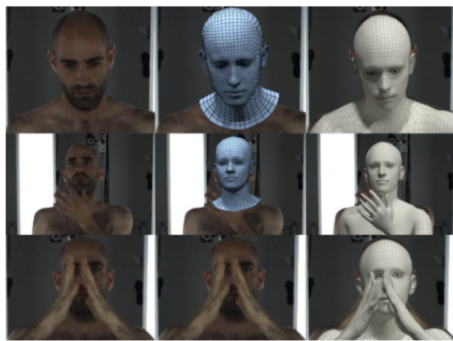
Figura 2.2: Esempio di ricostruzione con il metodo SMPL.

riesce a cogliere finemente dettagli e deformazioni di materiali morbidi (come la pelle e la carne umana), che si deformano in base alla posa del corpo e ai suoi cambiamenti: non tiene quindi in considerazione le correlazioni tra posa e shape. Per sopperire a questo problema in STAR viene definita una funzione correttiva per ogni giunto dell'albero cinematico di cui si compone la mesh, che utilizza sia un parametro per la body shape che per la posa. Inoltre, STAR è stato allenato su un dataset diverso rispetto a SMPL per avere più varietà di forme del corpo. Le mesh create possono, inoltre, essere animate velocemente e facilmente.

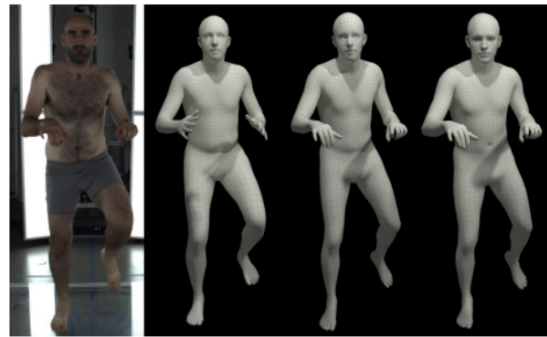
Diverse sono le estensioni implementate a partire da SMPL [12]:

- **DMPL** (Dynamic SMPL): basato sui vertici anziché sulle deformazioni triangolari per catturare in maniera più accurata le deformazioni dinamiche della pelle. Dal momento che queste dipendono per la maggior parte dalla forma del corpo rappresentata, DMPL viene testato utilizzando corpi di diverso BMI, imparando un modello delle deformazioni dinamiche che dipende dalla forma del corpo.
- **SMPL-X**: SMPL unito a **FLAME** (per il volto) e **MANO** (per le mani). Si tratta di SMPL più espressivo del precedente che permette di modellare e catturare finemente i dettagli e i movimenti di volto e mani a partire da immagini RGB.
- **SMPLify**: modello che stima le caratteristiche bidimensionali del corpo e ne ottimizza i parametri per adattarli alle caratteristiche.
- **SMPLify-X**: basato sullo stesso principio di SMPLify ma con l'aggiunta delle mani e del viso.

Dopo una panoramica sulla categorizzazione dei vari metodi di ricostruzione del corpo, ne verranno ora descritti alcuni esempi.



DMPL



SMPL-X

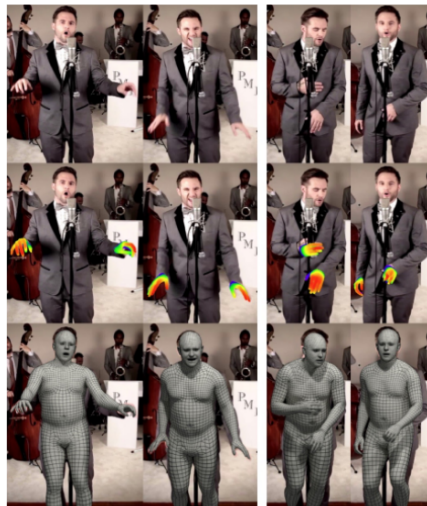


Figura 2.3: Esempi di implementazioni SMPL.

Tra i metodi che si concentrano sul migliorare la ricostruzione della shape dei modelli creati nonché sulla corrispondenza con le pose mostrate nelle immagini 2D in input, troviamo **Learnable mesh human** [14], che crea la mesh del corpo in input basandosi sui vertici del modello SMPL anziché sui giunti, per permettere di modellare pose e rotazioni degli arti più complesse. Vengono inoltre tenute in considerazione le oclusioni per evitare problemi di inconsistenza: per ogni vertice, infatti, viene calcolata la visibilità a partire dall'immagine in input. Anche **RoboSMPLX** [21] fornisce una ricostruzione della forma del corpo e della posa, nonché delle mani e del viso servendosi di tre sottoreti diverse e utilizzando il *contrastive learning* per stimare la posa e la forma di tutte le parti del corpo.

Altri metodi poi, si occupano di stimare solo la posa. È il caso di **Poseformer** [26], che riceve video in input, da cui estrae informazioni sul movimento del soggetto nello spazio sia nel dominio del tempo che della frequenza, e ne stima soltanto la posa dei giunti, senza occuparsi di ricreare una shape simile a quella in input. Per gli scopi di questo progetto di tesi, tuttavia, i metodi che si concentrano solo sulla ricostruzione della posa del corpo sono irrilevanti, dal momento che il nostro soggetto in input sarà posizionato in una posizione standard (di tali prerequisiti si parlerà diffusamente nel prossimo capitolo di questa trattazione).

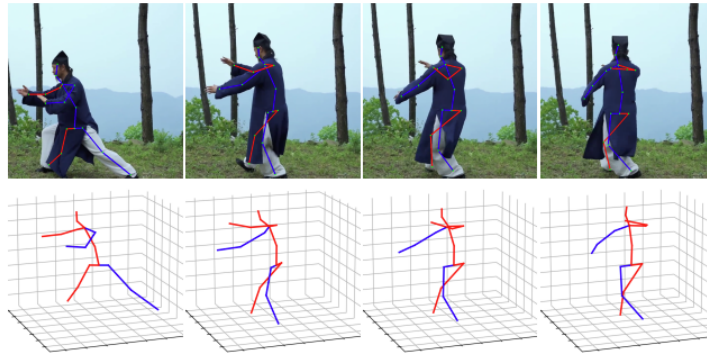


Figura 2.4: Stima della posa da parte del metodo Poseformer

2.2 Ricostruzione dei vestiti

Come già anticipato nell'introduzione a questo capitolo, la maggior parte delle ricerche di questa sezione fa riferimento allo stato dell'arte aggiornato a Maggio 2024. Nell'ultima parte di questa sezione verrà inoltre effettuata una breve panoramica sui progetti presenti e accessibili a Gennaio 2025.

A differenza della ricostruzione del corpo, quella dei vestiti risulta particolarmente difficile: la geometria dei capi di abbigliamento è molto complessa (per esempio per

quanto riguarda le pieghe dei tessuti) e ricostruire i dettagli fini richiede numerose misure e ad alta risoluzione, attraverso sistemi multicamera o fotocamere di profondità (le immagini RGBD infatti offrono l'opportunità di misurare i tessuti ad alta risoluzione al livello delle pieghe).

Tuttavia, la maggior parte dei metodi che si occupano di tale task utilizza in input una singola immagine RGB, con l'assunzione che la deformazione del tessuto risieda in uno spazio latente dove i modelli di base della forma possono essere appresi online. Questi metodi, però, modellano la deformazione dei vestiti in isolamento, in uno spazio in cui le interazioni tra il tessuto e la mesh del corpo non vengono prese in considerazione.

I metodi di ricostruzione dei vestiti si dividono in tre macrocategorie:

- **Physics Based Simulation (PBS)**

Vengono utilizzati per definire le deformazioni dinamiche dei capi di abbigliamento e utilizzano un gran numero di equazioni non lineari ad ogni step di ricostruzione. La pipeline dei metodi basati su PBS consiste nella progettazione e nel design dei capi di abbigliamento in due dimensioni, nel fare il grading degli stessi (ovvero la scalatura delle sue dimensioni), nell'adattare il vestito al corpo tridimensionale e infine nella regolazione dei parametri per fittare in maniera ottimale il tessuto alla mesh umanoide. Questo processo, tuttavia, può richiedere molto tempo anche a persone esperte nella ricostruzione. In generale sono necessarie simulazioni che coinvolgono un gran numero di triangoli per la modellazione e l'animazione dei capi, rendendo questi processi computazionalmente costosi.

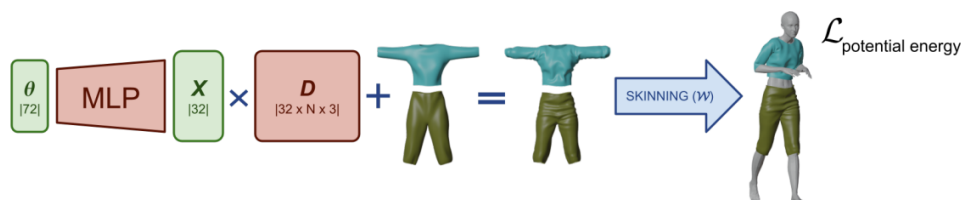


Figura 2.5: Rappresentazione della pipeline di ricostruzione per i metodi PBS.

- **Metodi data driven**

Si occupano di catturare ed estrarre i capi di abbigliamento a partire da immagini di umani vestiti, scansioni dinamiche, immagini RGBD, e farne il retarget su soggetti nuovi; la limitazione di tale processo è che la posa del corpo rimanga la stessa nonostante la sua forma cambi. Sviluppare ed utilizzare i modelli dipendenti dalla posa del corpo potrebbe essere una soluzione, che però richiede un gran numero di dati per il training e permette di ricostruire correttamente le pose del corpo solo se simili a quelle del dataset di allenamento. La maggior parte dei metodi più recenti si basa invece sul deep learning:

la variazione dello stile di abbigliamento avviene grazie uno sketch che l'utente fa tramite VAE (con posa fissa). Per un singolo stile, la posa e la forma vengono predetti utilizzando reti e modelli che elaborano sia il corpo che il capo di abbigliamento. Gli effetti della posa sui tessuti vengono modellati tramite *normal map* o *displacement map* nello spazio UV.

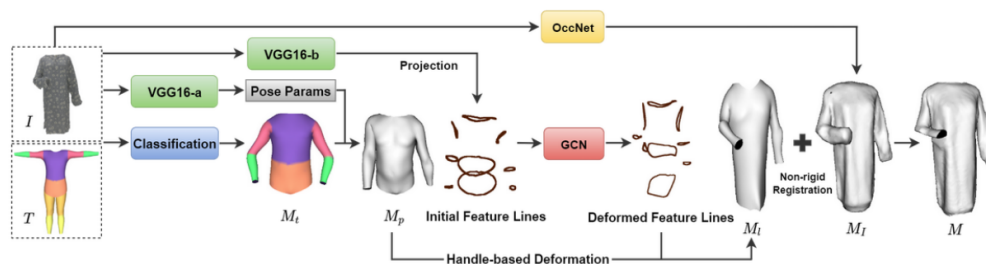


Figura 2.6: Rappresentazione della pipeline di ricostruzione per i metodi data-driven.

• Graph-based Methods

Vengono utilizzate delle *GNN* (*Graph Neural Networks*) per la modellazione basata sull'apprendimento di sistemi fisici dinamici. Tali metodi presentano tuttavia una propagazione lenta dei segnali attraverso il grafo e una difficoltà nella scelta del numero corretto di passaggi da effettuare per ottenere comportamenti e ricostruzioni ottimali. Infatti, un numero insufficiente di passaggi nella ricostruzione dei vestiti potrebbe condurre ad un eccessivo stiramento o a dinamiche innaturali. Diverse sono state le implementazioni per ovviare a questo tipo di problemi (come le architetture multilivello); tuttavia i metodi basati su grafi rimangono molto utili e funzionali dal momento che hanno dimostrato di poter modellare diversi tipi di sistemi fisici, inclusa la simulazione di tessuti.

Una seconda suddivisione dei metodi di ricostruzione separa quelli **impliciti** da quelli **espliciti**.

Per quanto riguarda i primi, questi rappresentano le superfici come un set di livelli di un campo scalare neurale implicito: i recenti metodi SOTA utilizzano la decomposizione dello spazio canonico o strategie basate su parti. A differenza dell'utilizzo di mesh predefinite, infatti, le superfici implicite non sono vincolate a particolari templates e possono modellare diversi capi di abbigliamento con topologia complessa. Sono tuttavia computazionalmente intensivi e difficili da addestrare con un numero limitato di dati.

La maggior parte dei metodi utilizzati, tuttavia, appartiene alla categoria di quelli espliciti, compatti ed efficienti, ideali per il rendering ma limitati a una topologia di dati fissi di uno specifico template. Per ovviare al problema dell'impossibilità di creare mesh di indumenti di topologia variabile, sono state introdotte le *point clouds*,

le nuvole di punti, che offrono flessibilità topologica e compattezza a discapito di un campionamento denso per la modellazione delle superfici; tale problema è stato risolto organizzando i punti in sottogruppi chiamati *patch*, che mappano lo spazio UV bidimensionale in quello 3D.

Dopo una breve panoramica delle diverse categorie di metodi che si occupano della ricostruzione dei vestiti, verranno ora analizzate nel dettaglio le caratteristiche e le problematiche più comuni a molti modelli di generazione di capi di abbigliamento.

Il codice latente degli indumenti è particolarmente efficace in fase di ricostruzione. La rete **DRAPENET** [36], per esempio, utilizza una rete di drappeggio, che permette di fittare e adattare correttamente i capi di abbigliamento su un corpo in input, servendosi del codice latente degli indumenti da utilizzare. Tale rete viene unita ad una rete generativa, composta da un encoder e un decoder. Nello specifico, l'encoder è allenato per comprimere i vestiti in input in codici latenti compatti, utilizzati come input dalla rete di drappeggio. Il decoder invece ricostruisce le mesh 3D dei vestiti a partire dal loro codice latente, permettendo così di modellare ed editare nuovi indumenti. Altri metodi, come **Clothwild** [44], invece sono progettati per ricavarsi direttamente il codice latente a partire da un'immagine RGB in the wild, senza necessità di reti o di encoder-decoder. Altri, poi, ricevono un grafo derivante dalla mesh dell'indumento a cui vengono aggiunti vertici e archi corrispondenti alla mesh del corpo (ad esempio **HOOD**) [39]: questi metodi permettono di raggiungere, andando a lavorare direttamente sui codici dei vestiti, dei risultati migliori e più accurati rispetto agli altri metodi SOTA, con la possibilità di andare a modificare direttamente gli indumenti in modo che si adattino alle forma del corpo della persona e alla sua taglia, e la possibilità di creare nuovi capi di abbigliamento e materiali.

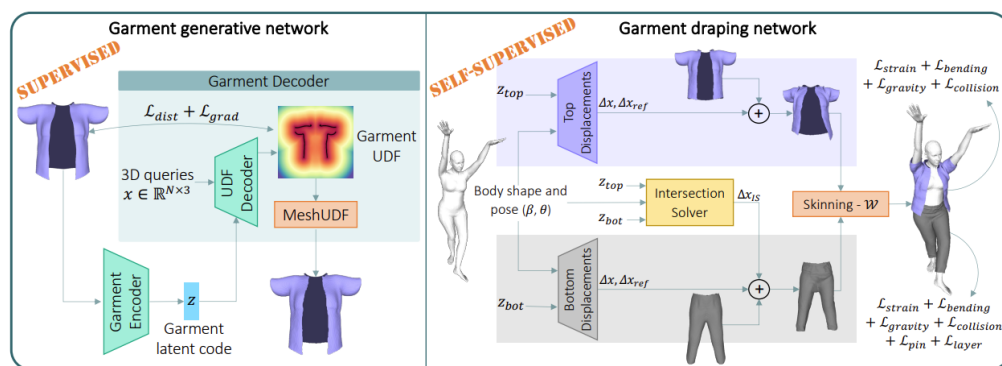


Figura 2.7: Pipeline di lavoro del metodo DRAPENET.

Alcuni metodi hanno difficoltà a riprodurre vestiti di alcune categorie in particolare (tra questi, **CloSET** [38], che fatica a modellare gonne e vestiti a causa della loro non uniforme distribuzione di punti se paragonati ad altri capi standard come t-shirt

e pantaloni) o capi di abbigliamento non presenti nella fase di training del modello (come nel caso di **PyMAF** [111], che invece permette di ricostruire perfettamente i dettagli fini dei vestiti già visti e le loro textures). La criticità più comune rimane, tuttavia, la modellazione delle scarpe, dal momento che hanno una geometria complessa e sono composte da un gran numero di vertici; per questo motivo molte reti non le ricostruiscono, lasciando il soggetto umano a piedi nudi.

Nel campo dell’animazione dei personaggi, poi, i metodi tradizionali utilizzano tecniche di rigging e skinning per mettere in posa i modelli umanoidi, ma non riescono a modellare realisticamente le deformazioni dei vestiti dipendenti dalla posa, come le pieghe e l’interazione tra indumenti e corpo. Per ovviare a tale problema ci si serve di una soluzione in due step: l’apprendimento di caratteristiche degli indumenti dipendenti dalla posa e la creazione di un dataset con modelli di abbigliamento realistici. Tuttavia, il retargeting di tali deformazioni nel dominio dei vestiti è un ambito non ancora studiato ad hoc. La forma dei vestiti può essere trasferita tra diverse forme corporee e pose, basandosi su corrispondenze 3D tra vestito e corpo o sulla deformazione manuale di un’immagine 2D. In molti casi, come nel già citato **CloSET** [38], si vanno a decomporre, dopo aver ricostruito la mesh 3D del capo di abbigliamento, le deformazioni dei vestiti in esplicite (proprie dell’indumento considerato e comuni a tutte le pose assunte dal corpo umano) ed implicite e dipendenti dalla posa in cui si trova il corpo sottostante (oltre che da quelle esplicite). In altri casi la suddivisione avviene tra statiche e dinamiche (come per **Neural Cloth Simulation** [42]), andando anche a studiare le proprietà dei materiali di cui sono composti i capi, al fine di ricostruire al meglio le relative mesh e nella maniera più realistica possibile. Tali metodi, tuttavia, utilizzano schemi di addestramento supervised che richiedono calcoli onerosi e dataset di grandi dimensioni.

Un’aspetto particolarmente delicato su cui concentrarsi è l’interpenetrazione tra la mesh del corpo umano e quella degli indumenti con cui viene vestito, o addirittura tra le diverse parti del capo di abbigliamento stesso nel momento in cui si vanno ad applicare deformazioni dinamiche al tessuto.

A tal proposito, esistono alcuni progetti che si occupano solo di deformare le mesh dei vestiti ricevuti in input, al fine di fittarli in maniera ottimale al corpo. È il caso di **DeePSD** [45], la cui capacità di animazione e deformazione può essere estesa anche ad outfit mai visti in fase di training, e può gestire diversi livelli di stratificazione dei vestiti, nonché di topologie e geometrie differenti. E’ qui opportuno citare anche **TailorNet** [33], che si occupa di fare il draping dei vestiti su un particolare tipo di corpo in una specifica posa: la limitazione di tale metodo tuttavia risiede nel fatto che si possono utilizzare solo un numero limitato di capi di abbigliamento e di pose e shape del corpo, imposte dalla rete. Se si prova infatti a fornire in input dei dati diversi di quelli su cui è stata allenata, il draping ricostruisce risultati visivamente e logicamente errati.

Inoltre, molti metodi non ricostruiscono le textures della mesh ma danno dei colori arbitrari ai capi di abbigliamento. Per risolvere tale problema vi sono delle reti specifiche che si occupano solo di estrarre la texture a partire da due foto di un indumento (frontale e posteriore) e di trasferirla correttamente sulla mesh (per esempio **Pix2Surf** [107]), automaticamente e in real-time. Un modello di questo tipo può essere molto utile per il rendering fotorealistico di esseri umani, la generazione di dati sintetici per l'apprendimento, il virtual try-on, l'arte e il design. Nonostante le numerose applicazioni, il trasferimento automatico delle texture di abbigliamento su indumenti 3D ha ricevuto pochissima attenzione. La stragrande maggioranza dei metodi recenti opera direttamente nello spazio immagine, trasferendo indumenti e textures da un'immagine in input ad un'immagine target, senza passare per lo spazio tridimensionale.

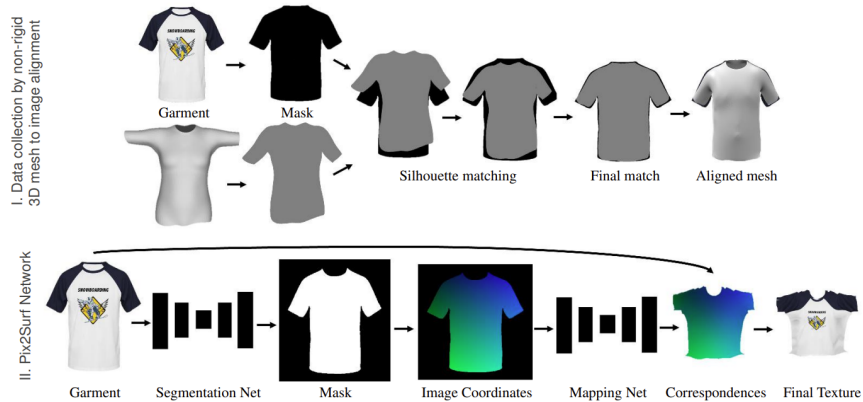


Figura 2.8: Pipeline di trasferimento della texture in Pix2Surf.

Molti progetti vengono allenati su dataset sintetici creati ad hoc e quindi faticano a ricostruire la mesh dei vestiti nel momento in cui come input hanno un'immagine in-the-wild.

Inoltre, la maggior parte dei metodi utilizza come modello umanoide su cui ricostruire gli indumenti quello generato con il metodo SMPL (lo stesso **ClothWild** [44], che si compone di due step, nel primo ricostruisce il corpo umano come SMPL e nel secondo crea i capi che andranno a vestire il corpo).

Per quanto riguarda il SOTA per la ricostruzione dei vestiti aggiornato a Gennaio 2025, si assiste ad un aumento esponenziale dei progetti che mirano a modellare capi di abbigliamento e textures sempre più accurati e dettagliati. La maggior parte di questi, infatti, richiede come input alla rete o più immagini multi-view del soggetto, inquadrato da diverse angolazioni per una precisione nella ricostruzione del modello a 360 gradi, o brevi video, in cui la/le persona/e si muovono e svolgono varie azioni, permettendo così di studiare anche la dinamica dei tessuti e come questi si adattano

al corpo. Non mancano gli input testuali che, servendosi di Large Language Models, permettono una ricostruzione text-to-mesh e un editing rapido dei vestiti.

Sempre meno reti, inoltre, si occupano soltanto della modellazione di vestiti: molte, infatti, ricostruiscono anche il corpo sottostante (alcuni creando due mesh separate, altri una unica). Tra i metodi del primo tipo si può citare **Ultraman** [106] che dispone di una funzionalità aggiuntiva per cui l'immagine in input viene descritta a parole prima di modellare la mesh (image-to-text).

Infine, la maggior parte dei metodi moderni, essendo appunto di ultima progettazione, è ancora in fase di revisione e di miglioramento (nonostante i risultati presenti nei paper e sui siti dedicati ai progetti mostrino ricostruzioni più accurate e meno errori rispetto ai metodi precedenti). Per questo motivo gli autori non hanno ancora pubblicato i codici di implementazione dei progetti, rendendo queste reti inutilizzabili per gli scopi di questo elaborato di tesi. Un'ulteriore curiosità e limite dei progetti più recenti che si occupano della ricostruzione dei soli vestiti ricevendo immagini in input e/o del draping: la maggior parte di queste reti riesce o a ricostruire solamente gli indumenti femminili o a deformare gli stessi per fittare correttamente solo il corpo femminile, escludendo quindi la possibilità di ricevere input in cui sono rappresentati uomini e aggiungendo un'ulteriore aspetto da considerare nel momento in cui si sceglie quale rete utilizzare.

2.3 Ricostruzione corpo e vestiti

Questa famiglia di metodi permette di ricostruire sia il modello del corpo che dei vestiti.

La maggior parte di questi progetti unisce le due mesh in una unica, utilizzando una rappresentazione olistica per modellare congiuntamente il corpo e l'abbigliamento a partire da immagini 2D, video 2D e mappe di profondità. Si incorre qui nell'impossibilità di separare le due mesh, il che rende questa sottocategoria inadatta ad applicazioni come il virtual-try-on e all'animazione delle mesh.

Progetti di questo tipo, sono, per esempio: **3DPeople** [110] che si basa su un dataset di umani vestiti e permette di stimare la forma del corpo e dei vestiti a partire da una singola immagine RGB, senza utilizzare modelli parametrici, e **SCANimate** [104], il cui obiettivo è quello di ricostruire un modello di umano vestito che sia semplice da deformare e da animare come lo sono quelli creati con il metodo SM-PL. Tale metodo deve supportare vestiti che si muovono e si deformano seguendo i movimenti e le pose del corpo, permettendo pieghe e drappeggi anche per posizioni che non sono ancora state utilizzate dal metodo (per il training o nella scansione di input). Come risultato si ha una ricostruzione realistica delle pieghe dei vestiti ma non dal punto di vista della texture, dell'albedo, della luce ecc.

Altri metodi si occupano invece di modellare separatamente il corpo e i vestiti, ma richiedono un dataset di grandi dimensioni di mesh vestite per il training, ottenibili tramite scanner 3D/2D o attraverso simulazioni fisiche. Uno di questi è **ClothWild** [44].

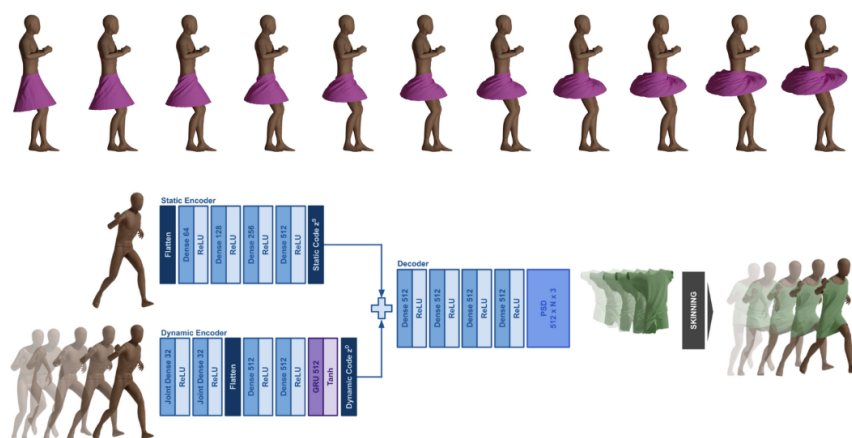


Figura 2.9: Metodo MGN per la ricostruzione delle animazioni.

Tra questi, un esempio è **SCARF** [32], il cui obiettivo è quello di creare ricostruzioni dettagliate e animabili degli umani attraverso SMPL-X e fare in modo che i vestiti possano essere trasferiti tra avatar diversi, ricevendo dei video in 2D come input alla rete.

samente maggiore rispetto all'utilizzo di reti che effettuino entrambe le ricostruzioni contemporaneamente.

2.4 Ricostruzione del volto

I primi lavori che hanno provato a ricostruire la mesh di un volto a partire da immagini risalgono al 1998, e precisamente dall'operato di Vetter and Blanz.

Oggi, la ricostruzione del volto viene utilizzata in moltissimi ambiti, tra cui la creazione di avatar 3D per AR/VR, l'editing di video, la sintesi di immagini, il riconoscimento facciale, il virtual make-up, le animazioni facciali guidate dal parlato.

Visto il grandissimo numero di applicazioni, questo argomento ha attirato molta attenzione, tanto che si sono sviluppati numerosi metodi per ricostruire una mesh del viso realistica in maniera ottimale e, soprattutto, che sia anche animabile (l'animazione avviene poi in uno step successivo, dal momento che le mesh fornite in output dalle reti non sono mai già riggate e quindi pronte per essere direttamente deformate).

Ci sono diverse problematiche e limiti nell'applicazione di tali metodi su larga scala. Primo tra tutti, la velocità di addestramento delle reti e del rendering, che richiedono ore o addirittura giorni per ottenere un singolo avatar: ciò limita l'ambito delle applicazioni e ostacola la creazione di esperienze immersive real-time. In secondo luogo, per ottenere ricostruzioni di alta qualità, i metodi attuali utilizzano rappresentazioni della mesh della testa che non sono compatibili con le pipeline grafiche standard, rendendo impossibile l'integrazione di tali asset all'interno delle scene. Inoltre, la maggior parte dei metodi genera avatar che possono essere renderizzati solo in un ambiente simile a quello in cui è avvenuta la scansione in input alla rete, dal momento che non separano la luce dalle proprietà intrinseche del materiale di cui il soggetto è composto.

L'obiettivo di questa tipologia di metodi è dunque quello di raggiungere l'efficienza (ovvero che il modello sia in grado di ricostruire identità e espressioni facciali mai visti durante il training), e l'alta fedeltà (al fine di preservare fedelmente i dettagli complessi presenti nell'immagine di input e di completare la mesh in modo realistico dove necessario (nel caso, per esempio, di occlusioni)).

I recenti metodi basati sull'apprendimento (*learning based*) sono in grado di raggiungere risultati realistici e di alta qualità nella ricostruzione di mesh 3D a partire da immagini 2D in input. Tali metodi possono essere suddivisi in due categorie principali (di cui si è già parlato diffusamente nel capitolo 1 di questa trattazione):

- **Model based**, basati sui modelli che costituiscono una base di partenza nella ricostruzione tridimensionale e utilizzano poche immagini 2D o su una sola.

Tuttavia, producono mesh che spesso mancano di dettagli geometrici e di accuratezza.

- **Model free**, basati su rappresentazioni discrete come voxel, mesh o nuvole di punti che permettono di ricostruire una maggiore varietà di forme del volto, a discapito però della bassa risoluzione delle generazioni in output.

Le limitazioni di entrambe queste tipologie di modelli sono state recentemente risolte attraverso le rappresentazioni implicite neurali, che rappresentano la geometria e l'aspetto del viso come un continuum, codificato nei pesi di una rete neurale, permettendo così di apprendere geometrie 3D complesse e ricche di dettagli direttamente dalle immagini 2D. Tuttavia, hanno come limite la necessità di un gran numero di viste in ingresso, in input alla rete.

Negli ultimi anni stanno avendo un grande successo i metodi chiamati **3DMM (3D Morphable models)**, che si occupano modellare caratteristiche proprie del volto (come la forma, l'espressione, l'albedo ecc.) attraverso coefficienti separati tra loro. Questi metodi, però, mentre sono perfettamente in grado di derivare e ricostruire la geometria generale del volto non modellano accuratamente i dettagli fini (come le rughe, le pieghe d'espressione, i pori della pelle ecc.). Per sopperire ad una tale mancanza, sono state introdotte delle *displacement maps* per la ricostruzione dei dettagli. Inoltre, sono estremamente dipendenti dalla risoluzione delle immagini che ricevono in input, che possono portare ad artefatti e a risultati non realistici se non soddisfano certi criteri base. I metodi *Shape by Shading (sfS)* apportano miglioramenti significativi per quanto riguarda la ricostruzione delle peculiarità facciali a partire da immagini e video monoculari, rimanendo però sensibili ad occlusioni o pose estreme. Tra questi, il progetto **HRN** [82], per esempio, si basa sul metodo 3DMM, andando poi a suddividere gerarchicamente il volto dettagli a bassa frequenza, a media e ad alta (questi ultimi, così come l'illuminazione e l'espressione, sono dipendenti dalla posa). Altri metodi invece si occupano di distinguere i dettagli dinamici di un volto (ottenuti interpolando le mappe di disposizione) da quelli statici (utilizzando i coefficienti del metodo 3DMM): i due tipi di dettagli vengono poi uniti insieme per creare l'output finale del modello. Un esempio di metodo che utilizza tale organizzazione è **HiFace** [79].

Per ricreare realisticamente un volto, le *wrinkle maps* giocano un ruolo cruciale: queste, infatti, sono mappe che memorizzano le normali della superficie del volto per espressioni “stretched” o “compressed” (gli estremi dello spettro possibile), ricavante da mesh ad alta definizione create manualmente da artisti del settore. Una *base normal map* memorizza i dettagli superficiali fini, come i pori della pelle, e il viso viene suddiviso in più regioni, ciascuna definita da una maschera specifica e memorizzata in una *texture map*. Per interpolare le wrinkle map di ciascuna regione e ricostruire quindi un volto il più realistico possibile, viene attribuito un peso scalare tra $[-1;1]$ per definire l'influenza di ciascuna mappa. I successivi miglioramenti

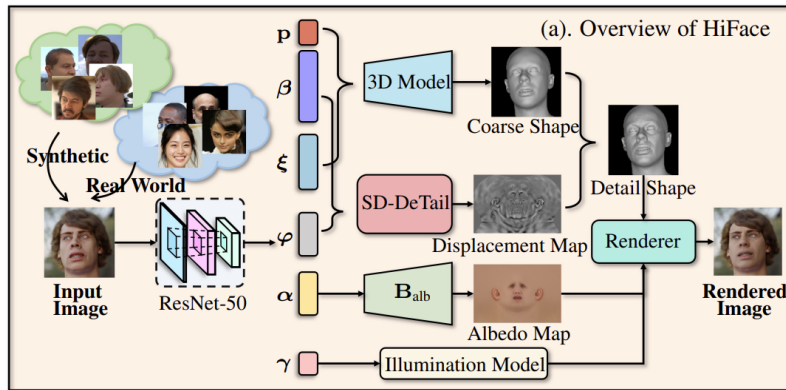


Figura 2.10: Pipeline per la ricostruzione del volto di HiFace.

per rendere questa tecnica utilizzabile in real-time hanno introdotto la generazione dinamica delle maschere di regione o dei pesi delle pieghe.

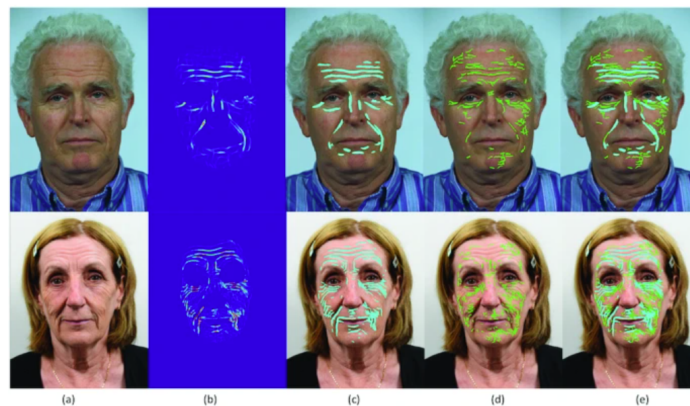


Figura 2.11: Esempio di wrinkle map detection.

Negli anni sono state introdotte diverse tecniche alternative per simulare le pieghe su superfici 3D, suddivisibili in: **simulazioni fisiche**, che trattano la pelle come una membrana elastica e modellano le deformazioni della stessa seguendo un metodo plastico lineare, e **simulazioni geometriche**, che ricostruiscono le pieghe usando primitive geometriche, come per esempio le Bezier cubiche.

Più recentemente, diversi metodi che si occupano della sintesi di espressioni facciali e texture della pelle hanno affrontato la ricostruzione delle rughe facciali attraverso approcci di machine learning (che utilizzano, per la maggior parte GANs).

Per quanto riguarda i dataset di allenamento delle reti, poi, nonostante i dati sintetici siano stati utilizzati in molteplici ambienti di ricostruzione 3D, la complessità della modellazione della testa li rende inadatti per la fase training. In letteratura,

infatti, la sintesi di tali dataset ha sempre richiesto semplificazioni e/o il focus su parti specifiche del viso, come la regione degli occhi, producendo quindi risultati lontani dall'essere realistici. Negli anni, tuttavia, si è cercato di rendere i dati sintetici il più realistici possibile, incoraggiando inoltre i modelli di apprendimento ad ignorare le differenze tra immagini sintetiche e reali; al contrario sono stati fatti pochi sforzi per migliorare la qualità dei dati sintetici stessi.

Nell'analisi dei metodi che si occupano della mesh del volto bisogna prestare attenzione se questi siano o meno in grado di ricostruire gli elementi "accessori" di un volto e non solo la sua forma base con dettagli fini ed espressioni facciali. La maggior parte dei metodi, infatti, non si cura della ricostruzione di accessori come occhiali, orecchini, barba e baffi, ma ignora tali elementi, andando a ricostruire automaticamente la porzione di viso che si trova sotto la loro superficie e che viene occlusa dalla loro presenza. È quindi complicato riuscire a trovare un progetto che sia in grado di modellare contemporaneamente tutte queste caratteristiche, insieme ovviamente al volto. Per citare alcuni esempi si ha: **PSAvatar** [96], che si concentra solo sulla ricostruzione dei facial hair e degli occhiali ma non degli altri accessori, **Ganfit** [100] e **Df2Net** [98] che sono in grado di modellare solo i facial hairs, **FLARE** [93] che invece non è in grado di ricostruire gli occhiali, al contrario di **MEGANE** [95] ecc.

Per quanto riguarda gli output poi, alcuni metodi modellano solo la mesh del volto senza il resto della testa: è il caso di **GOHA** [94], in cui l'output appare come una maschera senza alcun sostegno tridimensionale del resto del capo. La maggior parte dei progetti, tuttavia, ricrea tutta la mesh della testa, completa di eventuali accessori; alcuni di questi modellano addirittura anche i capelli. Per gli scopi della nostra tesi, tuttavia, si preferisce tenere questi due step separati, come già avviene per la modellazione del corpo e dei vestiti, dal momento che si ritiene che la ricostruzione dei capelli sia più accurata ed efficace se avviene con un metodo specifico e non insieme al volto.

Nell'ambito della modifica delle mesh tridimensionali generate, alcuni metodi sfruttano le capacità di editing avanzato delle GAN; tali metodologie, però, non possono essere utilizzate anche sulle mesh dinamiche. Altri metodi, invece, si concentrano sul trasferimento dello stile dell'avatar (utilizzando text prompt o immagini ad hoc), risultando tuttavia inefficaci nel preservare l'identità della persona rappresentata. A tal proposito, il metodo utilizzato nel progetto **Geneavatar** [98] introduce un nuovo approccio per l'editing delle mesh del volto, basato su un modello generativo che tiene conto delle espressioni del viso (*expression-aware modification generative model*) e che permette di traslare l'editing bidimensionale di una singola immagine in un campo di modifica a 3 dimensioni. Per garantire l'efficacia del progetto, sono state sviluppate diverse tecniche: l'individuazione delle modifiche del viso dipendenti dall'espressione, il miglioramento della convergenza del modello in uno spazio

latente implicito, e la strategia di perdita basata sulla segmentazione. Per ottenere delle texture dettagliate e precise.

Tale metodo può essere applicato a diversi avatar volumetrici basati su 3DMM e consente di renderizzare la mesh da nuovi punti di vista in modo coerente, effettuare modifiche dettagliate sulla stessa da più angolazioni e su più espressioni, preservare l'identità del soggetto durante il processo di editing.

Infine, per quanto riguarda l'animazione dei volti, sta diventando sempre più popolare l'utilizzo di **GANs**, che invece di limitarsi a ricostruirne la forma tridimensionale del viso, ne rappresentano il movimento (espressione e posa), andando matchare l'espressione in input alla rete con quella di un'immagine target, conferendo così movimento e animazione al viso (modello robusto anche a cambiamenti di posa significativi nel corso dell'animazione). Grazie alla capacità delle reti generative, questi metodi producono risultati ad alta fedeltà e con un aspetto più realistico rispetto ai metodi basati su modelli 3DMM. Anche il metodo NeRF ha mostrato buone prestazioni nella ricostruzione delle scene 3D, tanto che in si è provato ad utilizzarlo in diversi progetti e ambiti per la ricostruzione e l'animazione del volto umano. Questo modello, tuttavia, sebbene raggiunga risultati molto realistici e dettagliati, richiede migliaia di fotogrammi di un singolo individuo per la fase di training: un numero troppo elevato per renderlo un metodo efficiente nell'animazione, in cui la quantità di fotogrammi richiesti crescerebbe enormemente. Combinare GAN e NeRF, però, potrebbe rappresentare una soluzione semplice e veloce, consentendo di creare animazioni dinamiche a partire da immagini statiche multi-view.



Figura 2.12: Esempio del metodo GeneAvatar.

2.5 Ricostruzione dei capelli

La ricostruzione dei capelli è una parte critica nella modellazione di avatar umani: l'acconciatura di un soggetto in media consiste in 100.000 ciocche di capelli, la cui ricostruzione diventa difficile, lenta e onerosa. I sistemi e i metodi che sono stati sviluppati fino ad ora producono risultati soddisfacenti in linea di massima: per raggiungere risultati che siano il più realisticamente aderenti alla realtà i costi computazionali in termini di tempo e di hardware utilizzati crescono enormemente: è infatti quasi impossibile garantire una ricostruzione della chioma umana in real time.

Altro fattore da tenere in considerazione è quello delle occlusioni tra i capelli stessi, che causano non pochi problemi e artefatti nel momento in cui si va a ricostruire la mesh tridimensionale dell'acconciatura.

Inoltre, l'illuminazione diretta da una singola sorgente luminosa risulta complessa a causa dell'estrema sottigliezza dei capelli, che varia tra 15 e 200 μm . La pratica comune consisteva nel modellare una geometria semplificata a cui venivano applicate più textures per rappresentare più ciocche. Questo approccio risulta tuttavia piuttosto artefatto, richiede un notevole lavoro da parte degli artisti e la mesh semplificata è difficile da animare in modo realistico. Recentemente, i metodi stanno adottando il rendering basato su ciocche ma il problema dell'aliasing rimane significativo.

Per ovviare a tale problema, sarebbero necessari centinaia di campioni per pixel. Allo stesso modo, la visibilità della luce può essere valutata utilizzando tecniche di *Shadow Map*. Tuttavia, la composizione che utilizza il canale alfa richiede che i pixel siano processati in ordine dal retro al fronte dell'immagine, e la maggior parte delle tecniche di *Order Independent Transparency (OIT)* o sono molto costose o non offrono una qualità sufficiente per la complessità di profondità elevata dei capelli. Inoltre, le mappe di orientamento codificano orientamenti locali "privi di segno" e non catturano la direzione di crescita dei capelli, aggiungendo un'ulteriore ambiguità al processo di ricostruzione.

Inoltre, la maggior parte dei metodi non riesce a ricostruire chiome/acconciature troppo complesse se non erano presenti nel dataset di allenamento.

Ci sono tre tipologie principali in cui i metodi per la ricostruzione delle chiome possono rientrare:

- **metodi image-based**, che ricevono in input una o più immagini e restituiscono un modello tridimensionale dei capelli. A seconda del numero di immagini ricevute, si dividono in: **single-view** (che richiedono immagini da un solo punto di vista e non calibrate e per la loro semplicità stanno diventando molto popolari) e **multi-view** (con immagini da più pov, complessi setup degli hardware utilizzati, lunghi cicli di processamento e ambienti controllati: utili in ambienti professionali ma non sono metodi consumer friendly). Un esempio

di approccio multi-view è il metodo **NeuralHairCut** [52], che, oltre a ricostruire i capelli del soggetto, si dedica anche alla testa e alle spalle, andando a calcolare la dinamica e la fisica delle interazioni tra le ciocche e le parti del corpo rappresentate.

- **metodi data-driven**, che permettono di avere una prima conoscenza completa dell’acconciatura del soggetto e si basano su dataset sintetici per arrivare alla ricostruzione della mesh tridimensionale. Necessitano di un gran numero di dati e convergono lentamente a risultati accettabili, richiedendo una gran memoria. Questi metodi ricreano la mesh della chioma basandosi sull’immagine in input al sistema, ma non raggiungono tuttavia un alto livello di dettaglio o di accuratezza nella modellazione della mesh e portando talvolta a risultati strani (come per esempio l’inversione della direzione di crescita dei capelli e artefatti per la stima della profondità).
- **metodi che si servono di reti neurali e del deep learning**: questa tipologia di metodi catturando via via maggiore attenzione e sta apportando miglioramenti nel campo della ricostruzione dei capelli. Le CNN volumetriche usano reti 3D convoluzionali per generare mesh di capelli ma sono limitate dai costi computazionali della convoluzione 3D e dalla risoluzione volumetrica delle generazioni. Questi metodi, infatti, utilizzano un codice latente globale per rappresentare l’intera forma 3D, limitandosi così a geometrie semplici e sacrificandone il realismo. Una possibile soluzione sarebbe quella di concentrarsi sulle caratteristiche locali per rappresentare la geometria 3D corrispondente, anziché codificare direttamente l’intera forma in un codice globale. **HairGans** [59], per esempio utilizza delle GANS per ricostruire la struttura della mesh tridimensionale della chioma a partire da un’immagine 2D, sfruttando il potenziale delle GANs per ricreare la distribuzione di dati complessi, e rappresentandola dapprima come forma grezza e campo di orientamento dei capelli per guidare la sintesi, generando successivamente le ciocche tridimensionali. **HairNet** [55], invece, si serve di una CNN per una rappresentazione compatta e continua della geometria della chioma, integrando le collisioni e la visibilità per arrivare a risultati realistici.

Al tempo stesso, tecniche basate sulla ricostruzione gerarchica e sulla segmentazione possono essere utilizzate per aumentare la risoluzione, generando dettagli assolutamente realistici. È il caso di **AutoHair** [56], che compie una segmentazione dei capelli e definisce la direzione di crescita degli stessi, ottenendo così una mesh tridimensionale super dettagliata e realistica della chioma della forma definita e seguendo la mappa di direzione (dallo scalpo alle punte dei capelli).

Dall’altra parte invece, tecniche che si basano su nuvole di punti portano a ricostruzioni più ad alta risoluzione grazie alla loro rappresentazione non

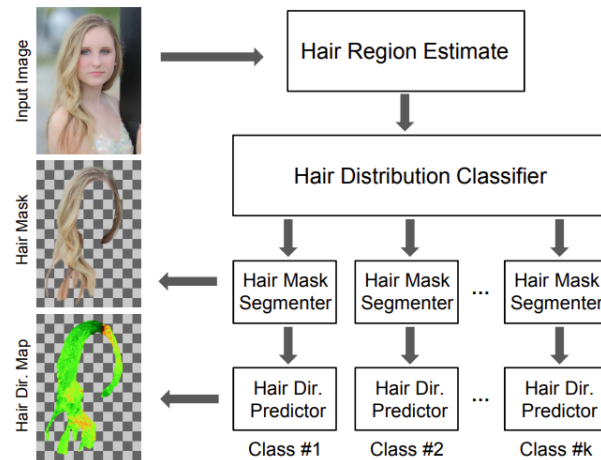


Figura 2.13: Segmentazione della chioma effettuata dal metodo AutoHair.

strutturata (non riescono però a catturare la struttura topologica delle ciocche di capelli che ricostruiscono).

Per quanto riguarda la rappresentazione e la modellazione della chioma, una seconda suddivisione può essere effettuata tra:

- **Rappresentazione basata su ciocche**

Rappresentazione geometrica costituita da un insieme di ciocche tridimensionali, dove ciascuna di queste viene parametrizzata da una sequenza di punti 3D connessi tra loro. A tale scopo, i metodi di cattura multi-view dei capelli ricostruiscono inizialmente una geometria di base a bassa risoluzione, utilizzando varie tecniche image-based di modellazione nello spazio tridimensionale. Successivamente, eseguono un processo di adattamento delle ciocche per ottenerne la mesh 3D connesse al cuoio capelluto. Questi metodi, tuttavia, spesso mancano di dettagli fini, come i capelli "volanti", a causa della bassa risoluzione della geometria di base.

- **Rappresentazione volumetrica**

Altra tecnica utilizzata per la modellazione dei capelli, in cui la chioma 3D viene ricostruita a partire da una singola immagine in input e utilizzando campi di orientamento volumetrico come rappresentazione intermedia, che può essere facilmente gestita da convoluzioni 3D. Tuttavia, a causa della bassa risoluzione della griglia dei voxel, i dettagli fini, come i capelli "volanti", non sono ben rappresentati.

L'animazione dei capelli, invece, risulta particolarmente difficile a causa della complessità dei movimenti e delle significative auto occlusioni tra ciocche. Generare dinamiche realistiche basandosi esclusivamente sul movimento della testa è altrettanto

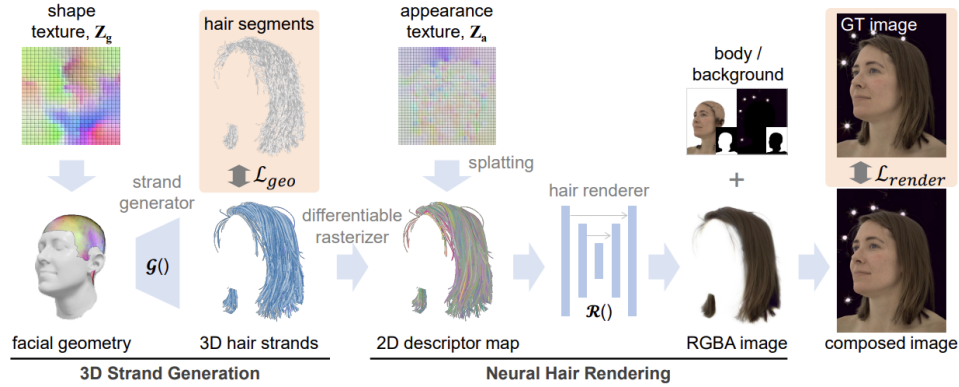


Figura 2.14: Esempio di modellazione basata su ciocche della rete Neural Hair Strands.

complesso, poiché il movimento dei capelli non è controllato solo dalla posizione della testa ma è anche influenzato dalla gravità e da forze inerziali. Dal punto di vista del controllo, inoltre, i capelli non rispondono in modo lineare alla posizione della testa.

2.6 Datasets

Ci si può servire di dataset per integrare dati con cui già sono stati allenati i metodi, per raggiungere maggior completezza e permettere di variare al massimo le ricostruzioni.

2.6.1 Dataset di corpi

Le maggiori problematiche che questi dataset devono affrontare sono, in primo luogo, la difficoltà di rappresentare un gran numero di tipologia corporee e di pose. Poiché i modelli umani, poi, vengono scansionati mentre indossano vestiti, le annotazioni 3D ottenute tramite fitting sono soggette ad errori. Infine, ottenere annotazioni separate per il corpo e l'abbigliamento è un compito piuttosto complesso. A tale scopo, sono stati creati dataset sintetici di mesh umane create attraverso SMPL-X. Il limite di tali dataset è che la loro natura sintetica li renda troppo “perfetti” per la ricostruzione di corpi reali.

È qui doveroso citare **Bedlam** [71], che si basa su dati sintetici derivanti da video ed include una grande varietà di forme corporee, tonalità della pelle e movimenti. Rispetto ai dataset precedenti, Bedlam utilizza corpi SMPL-X con capelli e abbigliamento realistico animati tramite simulazione fisica. Grazie al realismo e alla scala di Bedlam, è stato dimostrato che i dati sintetici sono sufficienti per addestrare regressori in grado di ottenere risultati soddisfacenti e in liane con gli altri metodi SOTA pur non utilizzando immagini reali durante il training.

In questo modo, i metodi allenati su questo dataset sono in grado di ricreare accuratamente e nel dettaglio sia la forma del corpo umano che la posa e in alcuni casi raggiungono risultati addirittura migliori. Il dataset si compone di video monocolori RGB di umani in movimento in formato SMPL-X, e ha come obiettivo il riconoscimento della topologia del corpo, delle azioni compiute e delle relative pose, rendendo più facile la ricostruzione di mesh tridimensionali rispetto ad avere in input immagini bidimensionali. Gli umani rappresentati nel dataset sono inoltre dotati di vestiti che si deformano in base alla posa del corpo, di capelli ecc. Simile a quello appena descritto è il dataset **SynBody** [70], in cui i corpi, di diverse forme, sono animabili e vestiti con strati di abbigliamento sovrapposti e sono dotati di annotazioni che permettono il supporto in attività e per funzionalità diverse.



Figura 2.15: Esempi di corpi in posa presenti nel dataset Bedlam.

2.6.2 Dataset di vestiti

La maggior parte di dataset che si stanno creando negli ultimi anni sono quelli composti da vestiti, che costituiscono la parte più complessa nella ricostruzione di un avatar 3D dal momento che la loro geometria varia per assecondare la posa assunta dal corpo sottostante. Creare questo tipo di dataset è diventata una necessità recente: per anni, infatti, ci si è occupati soltanto di creare dataset di corpi nudi, trascurando la complessità dietro la raccolta di dati per i vestiti.

Essi si possono suddividere in dataset **sintetici** (costituiti da dati più facili da raccogliere ma che talvolta portano a ricostruzioni delle mesh irrealistiche) e **scansionati** (raccolti attraverso diverse camere da più punti di vista, ottenendo scansioni non della migliore qualità possibile).

Inoltre, a seconda del dataset preso in considerazione possono comparire o meno le mesh del corpo sottostante (in alcuni casi esse vengono rimosse manualmente per garantire modelli si soli vestiti, come nel caso di **DeepFashion3D** [62]).

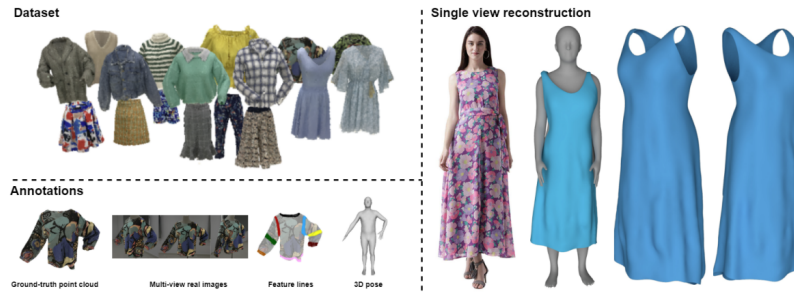


Figura 2.16: Esempio tratto dal dataset DeepFashion3D.

Dal momento che entrambe queste tipologie di dataset presentano delle criticità, nel corso degli anni sono stati sviluppati dei dataset sia con la mesh umanoide integrata che non, cercando di ovviare alle limitazioni di entrambi i modelli. I dataset ottimali, quindi, raccolgono rappresentazioni di capi d'abbigliamento che siano dotate di una fisica plausibile, la cui mesh sia separata da quella del corpo umano, e che presentino sequenze temporali di movimento del corpo per renderli adatti alla ricostruzione delle animazioni.

Un esempio è **CLOTH4D** [66], dataset composto da tre tipi di dati: mesh di abbigliamento dettagliate con texture accurate (tutti gli indumenti sono stati infatti creati manualmente e poi simulati, seguendo i principi del design e della moda) abbigliamento texturizzato separato dalla mesh del corpo sottostante, sequenze di movimento del corpo umano vestito (con un set di 289 azioni che ricoprono sia dinamiche di base sia posizioni più complesse).

Come ulteriore miglioramento il dataset **SIZER** [61] tiene anche in considerazione come il capo di abbigliamento vesta il corpo sottostante in funzione della misura e della taglia, dando quindi come input al metodo la forma del corpo e il vestito da indossare, segmentandone la geometria per adattarsi meglio alla mesh sottostante. Sizer è quindi un dataset di corpi umani che indossano lo stesso vestito in taglie diverse e di mesh 3D data driven ricavate da scansioni reali che rappresentano diverse forme del corpo e misure.

Esistono anche dataset che si occupano della sola segmentazione degli indumenti in modo da ricreare i vestiti con le giuste pieghe che fittino bene i corpi (presenti nel dataset di training): è il caso di **GIM3D** [67], un dataset sintetico di personaggi umani 3D vestiti e in diverse pose. La segmentazione 3D degli indumenti è un'operazione particolarmente complessa a causa dell'estrema varietà di forme che il capo di abbigliamento può assumere, anche all'interno della stessa topologia. Il dataset contiene, infatti, oltre 4000 modelli 3D prodotti tramite simulazioni fisiche di abiti realizzati con tessuti, taglie e livelli di aderenza differenti, utilizzando avatar umani animati con una grande varietà di forme corporee, ed etichettando ogni mesh creata per distinguere gli abiti e le parti visibili del corpo.

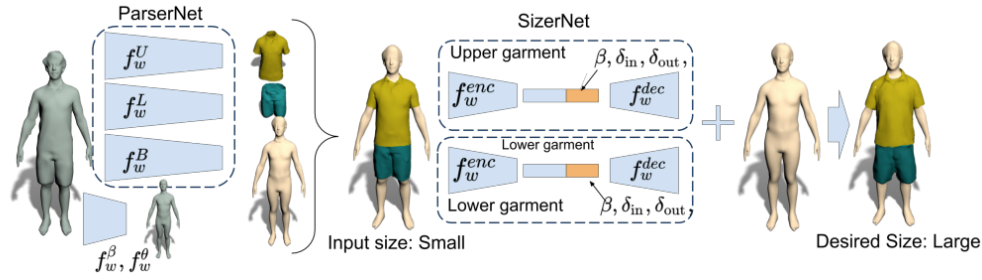


Figura 2.17: Pipeline di adattamento dei vestiti adottata da SIZER.

Nella maggior parte dei dataset di vestiti/accessori, inoltre, le scarpe non compaiono dal momento che hanno una geometria complessa dotata di molti vertici e che pochi metodi di ricostruzione del corpo le integrano.

2.6.3 Dataset di volti/espressioni

I dataset di questa tipologia si compongono di scansioni di volti animabili e riggabili, utili per permettere ai metodi che li utilizzano in fase di allenamento di imparare e predire le variazioni/deformazioni di piccole parti del volto causate dalle espressioni che il viso assume (per esempio le rughe). I dati presenti nei dataset vengono catturati in due modi principali: in maniera attiva, attraverso segnali ottici, e in maniera passiva attraverso l'utilizzo di fotocamere. Negli ultimi anni, la ricostruzione del corpo a partire da un'unica vista (*Single-View Face Reconstruction, SVFR*) si è sviluppata rapidamente: i metodi che si basano su questo tipo di ricostruzione possono essere divisi in **parametrici** e **non parametrici**.

I primi trattano la ricostruzione della forma del volto come un problema di adattamento ad un modello 3D parametrico, ottimizzando o prevedendo un vettore che rappresenta tale mesh a partire da un'immagine (spesso, tuttavia, non riescono a modellare i dettagli fini di un volto); i secondi riescono invece a modellare la geometria e le caratteristiche fini di un viso e ad alta frequenza.

Un esempio di dataset che mixa la rappresentazione parametrica e non è **FaceScape** [68], un dataset dettagliato di volti umani riggabili con una geometria dettagliata e in diverse espressioni.

Sono presenti anche raccolte di dati che non si focalizzano sulla forma del volto e sulle sue espressioni/movimenti, quanto piuttosto sulla ricostruzione delle sue textures: è il caso di **FFHQ-UV** [72], che contiene mappe UV di textures dei volti di alta qualità, integrate con illuminazione, espressioni neutre e regioni del volto "pulite" (senza accessori come occhiali, capelli ecc.): questo permette di renderizzare al meglio e realisticamente modelli tridimensionali in diverse condizioni di luce,

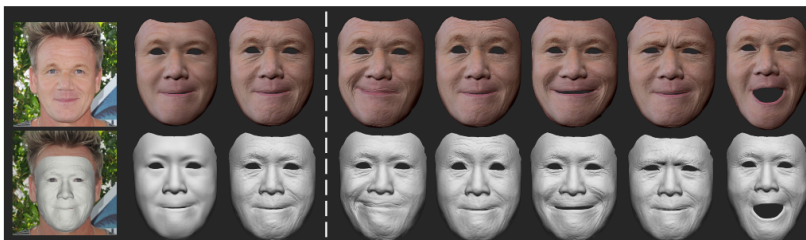


Figura 2.18: Dati presenti nel database FaceScape.

raggiungendo risultati più accurati.

Ci sono anche dataset che ricreano tutto il modello 3D della testa: è il caso di **DAD-3DHeads** [69], un dataset ampio, denso e diversificato che include annotazioni di oltre 3.500 punti di riferimento che rappresentano accuratamente la forma della testa 3D a partire da scansioni. Il modello di ricostruzione della mesh, **DAD3DNet**, addestrato su tale dataset, apprende parametri di forma, espressione e posa, eseguendo la ricostruzione 3D di una mesh FLAME.

2.6.4 Dataset di acconciature

Recentemente, sono stati creati diversi dataset dedicati alle acconciature, utilizzati soprattutto per l'addestramento di nuove reti e applicazioni per le industrie di bellezza e di moda, per i virtual try-on di chiome ed eventualmente per suggerire l'acconciatura adatta in base alla forma del viso e ad altre caratteristiche facciali. Tali dataset, tuttavia, presentano alcune limitazioni: in primo luogo la risoluzione nelle immagini di cui sono composte è solitamente bassa se confrontata con quella delle immagini scattate dagli utenti, causando risultati insoddisfacenti nei modelli; in aggiunta, il numero di immagini in questi dataset è spesso troppo limitato per consentire rappresentare correttamente una grande varietà di acconciature; infine, tutti i dataset esistenti modellano solo la chioma senza tenere in considerazione la forma del viso, i landmarks e altri attributi facciali.

Tra i dataset che si propongono di risolvere due di questi problemi, vi è **K-Hairstyle** [75], un dataset di grande scala che include: 500.000 immagini ad alta risoluzione (fino a 4032×3024 pixel), maschere di segmentazione delle chiome che includono etichette per definire vari attributi, tra cui lunghezza, colore e altre caratteristiche demografiche della persona rappresentata nell'immagine (per esempio, età e genere), categorie di acconciature popolari (dal momento che tale dataset è stato creato in Corea, i modelli di acconciature saranno per la maggior parte quelli che rispecchiano la cultura e la moda locale). Le applicazioni principali sono: la tintura virtuale dei capelli, il trasferimento di acconciature da un'immagine 2d ad un'altra, la classifi-

cazione di tali acconciature.



Figura 2.19: Tintura virtuale per immagini del dataset K-Hairstyle

Per risolvere, invece, anche il terzo problema tra quelli evidenziati, è stato introdotto **CelebHair** [76] (che si basa sul dataset di volti *CelebA*), un dataset su larga scala creato con l'obiettivo di proporre consigli di acconciature. CelebHair si compone quindi di oltre 200000 immagini facciali (derivanti, appunto, da CelebA) con le relative acconciature e una vasta gamma di attributi facciali.

Tra questi ultimi vi sono quelli ereditati da CelebA (come la curva delle sopracciglia, gli occhiali, gli zigomi, l'età, la rotondità del viso, il genere ecc.), attributi calcolati dai landmarks facciali attraverso algoritmi di rilevamento (tra cui, l'altezza della fronte, la lunghezza e lo spessore delle sopracciglia, la larghezza e la lunghezza degli occhi, la distanza pupillare ecc.), la classificazione della forma del viso e delle acconciature (utilizzando una rete CNN).

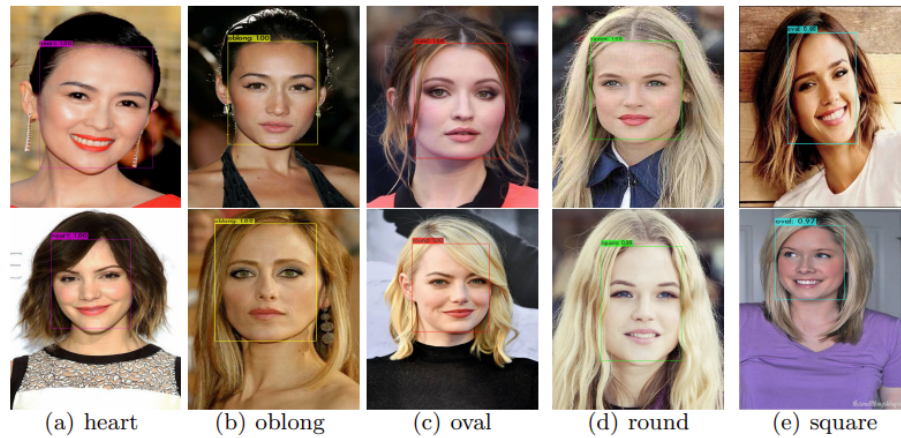


Figura 2.20: Immagini estratte dal dataset CelebHair.

Capitolo 3

Strumenti utilizzati

3.1 Prerequisiti

Prima di andare ad analizzare nello specifico la tecnologia utilizzata per questo progetto di tesi, è opportuno concentrarsi sui prerequisiti del progetto nonché sulle motivazioni che hanno portato a preferire l'utilizzo di una rete piuttosto che di un'altra.

Tali assunzioni sono:

- la ricostruzione di un avatar tridimensionale non è soggetta a vincoli di tempo, nel nostro caso specifico: tutti i metodi, infatti, che richiedono lunghi tempi di elaborazione e di rendering della mesh sono stati comunque presi in considerazione.
- l'input sarà una singola immagine RGB monoculare in formato verticale, scattata da un qualsiasi tipo di dispositivo elettronico.
- il soggetto rappresentato può trovarsi in-the-wild.
- la persona dovrà essere ritratta posizione standard e a figura intera: lo scopo di questa tesi, come già stato accennato nel capitolo precedente, non è infatti quella di ricostruire pose complesse bensì di modellare una mesh umanoide che nella sua forma sia quanto più simile all'originale in input.
- vi è necessità di separare la mesh del corpo umano da quella dei capi di abbigliamento con cui è vestita, per permettere la personalizzazione di questi ultimi.
- non sono necessari particolari parametri o coordinate di posizionamento della camera.

- l'interazione dell'avatar 3D con l'ambiente circostante (ed eventualmente con altre mesh umanoidi) viene ignorata, così come la gestione delle occlusioni tra soggetti diversi e parti dell'environnement in cui si trovano i soggetti.
- le reti che ricostruiscono più parti della mesh finale (come per esempio corpo e vestiti, corpo e testa, testa e capelli) non hanno ricevuto molta attenzione: si prediligono infatti metodi che tengono separato ogni step della pipeline, permettendo di raggiungere un livello più elevato di dettaglio e di qualità dell'output.

Basandosi su queste considerazioni, sono stati selezionati ed implementati per i nostri scopi i metodi di seguito descritti.

3.2 Reti e dataset

3.2.1 Corpo

Per quanto riguarda la modellazione del corpo, è stata utilizzata la rete **LVD (Learned Vertex Descent)** [20]. Tale rete è stata selezionata del momento che si concentra soprattutto sulla ricostruzione shape del corpo piuttosto che sulla sua posa, a differenza della maggior parte dei metodi SOTA, combinando i vantaggi dell'ottimizzazione classica (in cui si cercano iterativamente i parametri del modello che meglio si adattano all'immagine in input, come keypoint 2D e silhouettes) e dei metodi di regressione basati sui dati (che sfruttano reti neurali profonde per prevedere direttamente i parametri del modello a partire dall'input). Il metodo converge all'output in real-time, richiedendo solo sei interazioni.

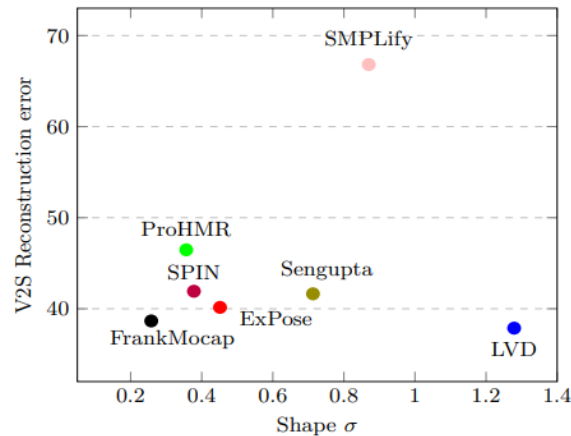


Figura 3.1: Confronto tra i metodi di ricostruzione del corpo.

PROCESSING

La rete riceve in input due immagini: una in cui è rappresentato il soggetto a figura intera, a cui viene rimosso il background e sostituito con uno sfondo nero, e una in cui il soggetto viene convertito in una maschera bianca, sempre su background nero. Il corpo modellato viene rappresentato tramite una mesh 3D, la cui topologia segue quella del modello SMPL, con 6890 vertici connessi da 13776 facce triangolari. A differenza di SMPL, tuttavia, LVD opera direttamente sui vertici, il che rende tale metodo applicabile anche ad altri modelli (come, per esempio, la ricostruzione delle mani) e utilizza 300 parametri per descrivere la shape del corpo (invece dei 72 di SMPL), permettendo così alla rete di rappresentare un maggior numero di forme corporee differenti.

METODO

1. Estrazione delle features locali dall'immagine in input.

Le feature locali vengono estratte dall'immagine attraverso un encoder-decoder a clessidra, addestrato da zero. Dato un vertice $v_i^t = (x_i^t, y_i^t, z_i^t)$ e l'immagine di input I , le feature locali vengono stimate tramite:

$$f : (I, \pi(v_i^t), z_i^t) \rightarrow f_i \quad (3.1)$$

dove $\pi(v)$ rappresenta una proiezione prospettica del vertice sul piano dell'immagine. La funzione $f(\cdot)$ viene condizionata dalla profondità z_i^t per generare le caratteristiche *depth-aware* (sensibili alla profondità).

2. Predizione dello spostamento dei vertici.

Le features locali estratte al passaggio precedente sono utili per prevedere lo spostamento dei vertici, permettendo di migliorare significativamente la geometria del modello seppur utilizzando un dataset limitato. Tale approccio differisce da quello tradizionale in cui l'intera immagine veniva mappata sui parametri globali della forma, determinando una scarsa corrispondenza tra immagine in input e mesh in output.

3. Implementazione della funzione $g(\cdot)$

Viene utilizzata una funzione $g(\cdot)$ che, dato il posizionamento corrente del vertice nello spazio 3D e le feature dell'immagine mappata nella sua proiezione 2D, prevede il modulo e la direzione del vettore che rappresenta la "discesa" più ripida verso la posizione corretta del i -esimo vertice, $(\hat{v}_i)$. Formalmente:

$$g : (v_i^t, f_i) \rightarrow \Delta v_i \quad (3.2)$$

dove $\Delta v_i \in R^3$ è un vettore con origine in v_i^t e punto finale in $\hat{v}_i$ (la posizione target). Il problema di ricostruzione consiste quindi nel ripetere Eq. (2) fino

alla convergenza di Δv_i .

Per implementare la funzione $g(\cdot)$ viene utilizzata una **MLP (Multi-Layer Perceptron)** a 3 strati, che riceve quindi in input: la stima attuale di ogni vertice v_i^t , la sua feature locale F-dimensionale f_i e ne prevede lo spostamento Δv_i per aggiornare la posizione del vertice.

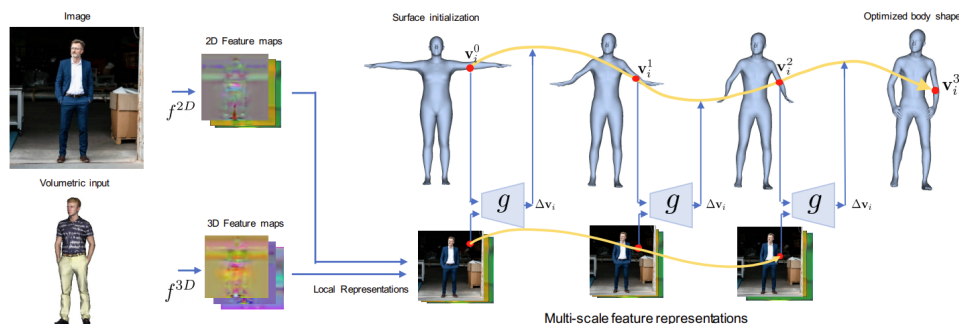


Figura 3.2: Pipeline della rete LVD.

LIMITI

La maggior parte dei limiti di tale rete derivano dai dati utilizzati nella fase di training. Per quanto riguarda la stima della forma corporea data una singola immagine in input, il modello è vincolato al modello SMPL e alle shapes presenti nei dataset di allenamento, il che porta LVD a non riuscire a ricostruire correttamente forme corporee diverse da quelle più comuni (come nel caso di una donna in gravidanza). Un ulteriore limite consiste nella scarsa diversità nelle pose di addestramento: se particolarmente complesse o rare portano a predizioni errate, in cui la ricostruzione non si adatta perfettamente all'immagine di input. Tuttavia, le parti del corpo stimate in modo errato vengono evidenziate con un colore che indica un alto livello di incertezza (in blu scuro), fornendo un'indicazione visiva degli errori.

3.2.2 Vestiti

Per quanto riguarda la ricostruzione dei vestiti, i requisiti richiesti alla rete erano, come già anticipato nel capitolo precedente, che ricostruisse i soli vestiti data una singola immagine in-the-wild come input. Tuttavia, dal momento che la mesh del corpo viene creata in un altro step della pipeline, utilizzare una rete per creare i soli vestiti senza nessun riferimento alla mesh del corpo sottostante se non l'immagine in input, può risultare piuttosto rischioso, portando a risultati irrealistici e/o a interpenetrazioni tra mesh.

A tal proposito era stata selezionata una rete, **ClothWild** [44], che permette di ricostruire anche il modello umanoide SMPL, in una mesh divisa da quella dei vestiti. In questo modo era possibile sostituire tale corpo con quello più accurato e

creato ad hoc da LVD. Tuttavia, la rete stessa, nonostante da paper e da progetto sembrasse assolutamente idonea ai nostri scopi, restituiva mesh dei vestiti piuttosto errate (soprattutto per quanto riguarda la lunghezza dei diversi capi di abbigliamento, dai pantaloni alle maniche delle maglie ecc.), come si può vedere nelle immagini sottostanti.

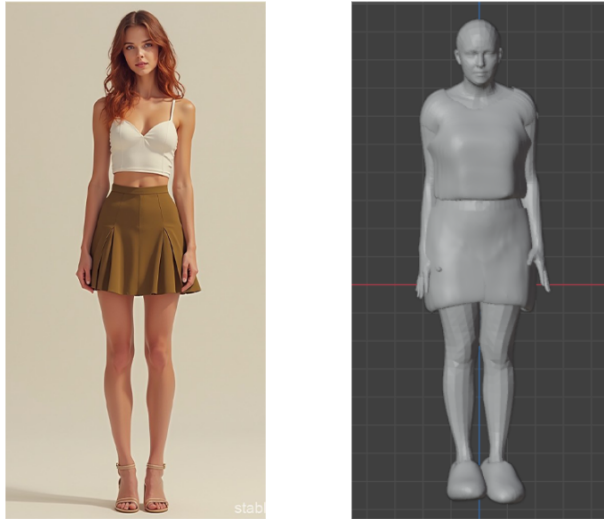


Figura 3.3: Esempi di ricostruzione dei vestiti con Clothwild.

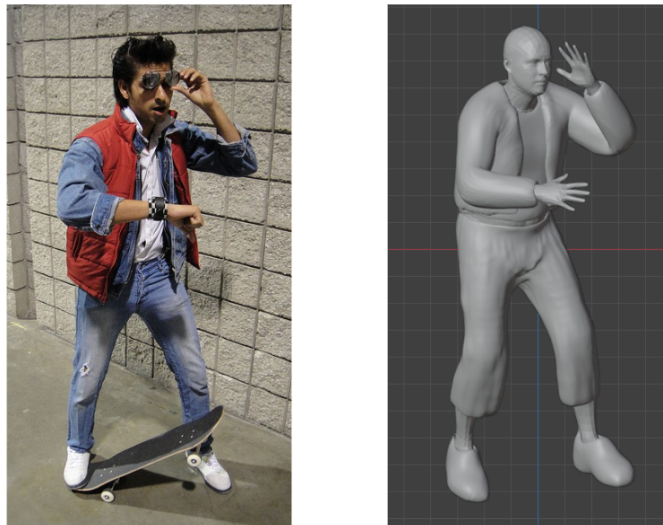


Figura 3.4: Esempi di ricostruzione dei vestiti con Clothwild.

Alla luce di tutte le limitazioni e delle problematiche evidenziate nell'analisi dei progetti SOTA, si è deciso di cambiare approccio, lasciando da parte le reti neurali per prediligere, invece, un metodo algoritmico.

Inizialmente si era pensato di servirsi di un tool online, **AI Describe Tool** [3], che analizza l'immagine in input al progetto e ne fornisce una descrizione dettagliata, identificando gli oggetti presenti e quindi i capi di abbigliamento con cui il soggetto è vestito. AI Describe Image permette di analizzare una foto, estrarre testo, descrivere una persona nell'immagine o generare un prompt per un'AI text-to-image. Tale tool era l'unico strumento che risultava funzionale per questo progetto di tesi, dal momento che la maggior parte dei progetti esistenti e installabili si occupa principalmente di text-to-image, il processo opposto rispetto a quello di cui si necessita in questo step. Per quanto riguarda, invece, i progetti del tipo image-to-text tutti quelli individuati permettono di descrivere l'immagine che ricevono in input solamente se corrisponde ad una di quelle precedentemente analizzate nella fase di training e presenti nel loro dataset. Con queste premesse era dunque impossibile servirsi di progetti/reti esistenti basati su LLM.

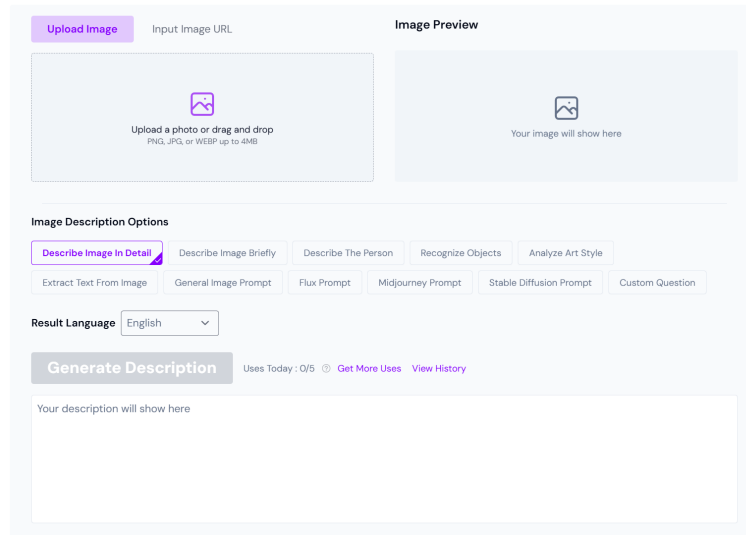


Figura 3.5: Interfaccia del tool AI Describe Image.

Il metodo attuale, invece, si compone dei seguenti due step:

- una prima fase in cui l'utente dovrà scegliere manualmente, attraverso un menù a tendina, quali vestiti dovrà indossare l'avatar e il colore da applicare a questi ultimi. All'interno del progetto sono infatti presenti diversi capi di abbigliamento, alcuni modellati a mano, altri importati nell'ambiente e opportunamente modificati;
- in base alla scelta effettuata dall'utente al punto precedente si procede a posizionare sul corpo, già presente nell'ambiente, il capo di abbigliamento corrispondente, a cui viene assegnato il colore selezionato tra quelli disponibili. Il processo appena descritto avviene grazie ad una serie di script che sono stati

implementati e integrati nel progetto Blender.

Nello specifico, si assiste a due tipi diversi di scripting. Per i capi di abbigliamento modellati a mano, la pipeline è la seguente: la mesh, rappresentata come *sew pattern*, viene opportunamente scalata e posizionata sul corpo del soggetto, viene attivata una *cloth simulation* per permettere all'indumento di adattarsi alla mesh umanoide, e, infine, si ha una fase di ottimizzazione in cui vengono applicati gli opportuni *modifiers* sul vestito appena modellato per renderlo il più verosimile e il più gradevole alla vista possibile. Per i capi di abbigliamento importati, il processo è simile a quello appena descritto ad eccezione della cloth simulation: in questo secondo caso si ha un maggiore pericolo di incorrere in interpenetrazioni tra le due mesh, ma le modifiche iniziali e i controlli tramite script permettono di eliminarne la maggior parte.

Al termine di questo step della pipeline si avrà quindi la mesh umanoide del soggetto opportunamente vestita e con le textures applicate agli indumenti.

3.2.3 Viso

Ricevendo un'immagine singola in-the-wild come input, **DECA (Detailed Expression Capture and Animation)** [88], permette, attraverso la regressione, di ricostruire la mesh tridimensionale del volto e i dettagli del viso, specifici per ogni individuo e che variano a seconda dell'espressione (particolare attenzione viene dedicata alle rughe). La parametrizzazione di tali dettagli consente poi di animare gli stessi, controllando i parametri di espressione e di posa della mandibola e sintetizzando nuove rughe mantenendo invariati i dettagli specifici del soggetto. Il regressore è stato inoltre addestrato per estrarre dettagli, forma, albedo, espressione, posa e parametri di illuminazione dall'immagine target.

PROCESSING

L'idea che sta alla base della rete consiste nel fatto che il volto di un individuo presenta due tipi di dettagli facciali: quelli sintetici specifici della persona e quelli dinamici dipendenti dall'espressione. DECA deve quindi essere in grado di differenziare queste due categorie, per ricavare i dettagli facciali sia dallo spazio latente dei dettagli specifici della persona sia dallo spazio delle espressioni.

METODO

1. Ricostruzione “grossolana”

L'immagine 2D in input viene codificata in un codice latente, la si decodifica per sintetizzare una seconda immagine bidimensionale e infine viene minimizzata la differenza tra l'immagine in input e quella sintetizzata. Nello specifico viene allenato un codificatore che estragga i seguenti parametri dal target: di forma grezza, di espressione, di posa, coefficienti di albedo, parametri della fotocamera e parametri di illuminazione.

2. Ricostruzione dei dettagli

La ricostruzione dei dettagli arricchisce con una mappa UV la geometria “grossolana” creata allo step precedente. Anche in questo caso si addestra un codificatore per codificare l’immagine in un codice latente a 128 dimensioni, che rappresenta i dettagli statici e specifici del soggetto rappresentato nell’immagine in input. Tale codice viene successivamente concatenato con parametri d’espressione (che catturano i dettagli dinamici delle rughe) e i parametri di posa della mandibola di FLAME. Viene poi trasmesso al decodificatore e convertito in una *normal map* per il rendering, a cui si aggiungono i parametri di illuminazione e di albedo.

3. Separazione dei dettagli

Viene introdotto un modello di spostamento dipendente dall’espressione che arricchisce FLAME con dettagli a frequenza media, fondamentali per migliorare la fedeltà visiva della mesh e per permettere una maggiore personalizzazione e un’animazione realistica delle ricostruzioni facciali. Tuttavia, è comunque necessario separare i dettagli statici da quelli dinamici: a tal proposito ci si serve di due immagini della stessa persona, in cui il soggetto ha quindi la stessa geometria grezza e gli stessi dettagli statici. La combinazione tra i parametri di posa della mandibola e di espressione dell’immagine 1 e il codice di dettaglio dell’immagine 2 permette di stimare i dettagli delle rughe del soggetto. In questo modo i dettagli ricostruiti possono essere animati in modo realistico, mantenendo tuttavia invariata l’identità della persona delle immagini.

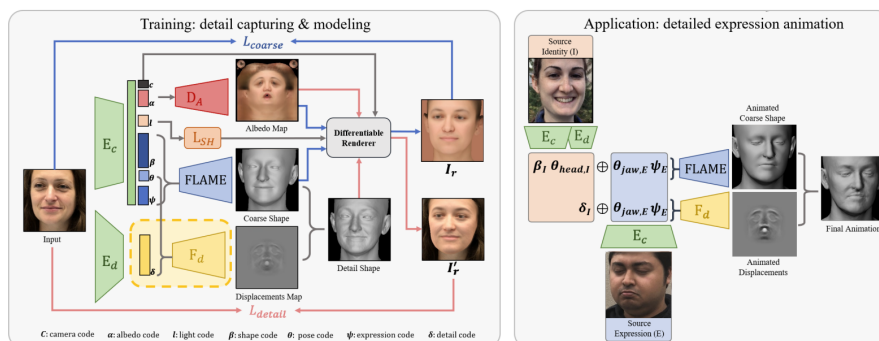


Figura 3.6: Pipeline di ricostruzione di un volto in DECA.

LIMITI

Sebbene DECA sia un metodo molto efficace per la ricostruzione della mesh del volto, e che, grazie alla sua accuratezza e velocità sia un modello assolutamente adatto per modellazioni real time, presenta tuttavia delle limitazioni:

- la qualità del rendering è limitata dal modello di albedo: vi è necessità, infatti, che non vi siano ombreggiature e ombre marcate e riflessi al fine di separare correttamente l'albedo dai dettagli geometrici.

- DECA non modella esplicitamente i peli facciali, causando distorsioni nel tono della pelle e deformazioni geometriche.
- sebbene DECA sia robusto con occlusioni comuni, fallisce in caso di pose estreme della testa, illuminazioni estreme, occlusioni significative (es. mani che coprono il volto).
- i dataset di addestramento contengono molte immagini a bassa risoluzione, che se da una parte migliorano la robustezza del modello dall'altra introducono dettagli rumorosi.
- **DECA-HR** (modello ad alta risoluzione): ha come punto a favore il fatto che aumenta la qualità visiva e riduce il rumore nei dettagli ricostruiti, ma è meno performante se riceve immagini a bassa risoluzione, con pose estreme o con espressioni particolari.
- DECA non include la lunghezza focale, limitando la capacità di ricostruire geometrie 3D da selfie: si tratta quindi di un modello a camera prospettica debole.

3.2.4 Capelli

Per la ricostruzione della mesh dei capelli è stata utilizzata la rete neurale **HairStep** [57], che si occupa di modellare la chioma a partire da un'immagine single-view e in-the-wild in input. Per questo motivo HairStep differisce dagli altri metodi che utilizzano invece solamente mappe di orientamento (soggette al rumore incerto e che non permettono una ricostruzione accurata e aderente alla realtà).

PROCESSING

Per il training della rete è stato creato un dataset ad hoc di 1250 acconciature tratte da immagini reali, con due tipi di annotazioni diverse:

- le mappe delle ciocche (**HiSa**): al fine di generare mappe di orientamento aderenti all'immagine in input, queste vengono create tracciando curve vettoriali 2D dense e ben allineate lungo i capelli, permettendo quindi di descrivere la crescita dei capelli a livello di pixel e rappresentandone la direzione a 360 gradi.
- le mappe di profondità (**HiDa**), che contengono solo informazioni geometriche, aiutando a ridurre il divario di dominio tra le immagini sintetiche e reali.

METODO

A partire dall'immagine target, vengono estratte due mappe: una delle ciocche e una di profondità. La corrispondente rappresentazione HairStep dell'input sarà dunque la seguente: $H=O;D$, dove:

- O rappresenta la mappa delle ciocche: un'immagine RGB con dimensioni $W \times H \times 3$, dove W e H sono rispettivamente la larghezza e l'altezza dell'immagine target.
- D rappresenta la mappa di profondità, e ha dimensioni $W \times H \times 1$ dove viene rappresentata per ciascun pixel la distanza minore tra i capelli e la fotocamera nella coordinata della fotocamera. Il metodo di stima della profondità è adattativo al dominio al fine di ridurre al minimo il divario tra le mappe di profondità dei dati reali e quelli sintetici.
- il colore in un determinato pixel xx nella mappa delle ciocche è definito come

$$O(x) = (M(x), O_{2D}/2 + 0.5)$$

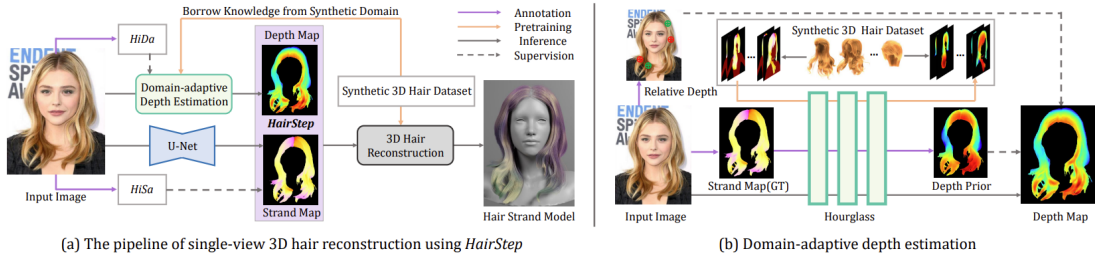


Figura 3.7: Pipeline di ricostruzione dei capelli di HairStep.

Al termine dell'estrazione delle mappe, queste ultime vengono combinate andando a creare una rappresentazione ibrida che viene quindi inserita in una rete per apprendere il campo di orientamento 3D delle ciocche e il campo di occupazione 3D di capelli nello spazio canonico di uno scalpo standard. L'output di Hairstep viene fornito ad una seconda rete, **NeuralHDHair**, che prevede i campi impliciti di occupazione e orientamento che rappresentano il modello tridimensionale dei capelli target. Al termine di questo processo, vengono sintetizzate le ciocche adottando il metodo di crescita descritto a partire dalle radici dei capelli dello scalpo.

LIMITI

Hairstep può fallire sulla ricostruzione di alcune acconciature rare e complesse, dal momento che la rete si basa su dataset sintetici che per loro natura presentano una quantità e una diversità limitate.

3.2.5 Segmentazione

Inizialmente, era stato individuato SATR come metodo per la segmentazione del corpo, con l'obiettivo di individuare la regione della testa della mesh umanoide creata allo step di ricostruzione del corpo, in modo da andarla a sostituire con la mesh

della testa creata ad hoc.

Il metodo **SATR** (**Segmentation Assignment with Topological Reweighting**) [27] permette di stimare una mappa di segmentazione 3D a partire da previsioni di bounding box in due dimensioni che sfruttano le proprietà topologiche della mesh 3D sottostante. Si compone di due parti: di **GLIP**, per stimare i riquadri di delimitazione 2D da diverse angolazioni delle fotocamere (che tuttavia fornisce solo una guida approssimativa per la segmentazione 3D e rischia di includere facce della mesh non correlate nel segmento target); di tecniche sviluppate per affinare la segmentazione, tra cui la ricalibrazione geodetica gaussiana (*gaussian geodesic reweighting*), che esegue una ricalibrazione robusta delle facce basandosi sulle loro distanze geodetiche dal centro del segmento potenziale preso in considerazione, e un *visibility smoothing*, che utilizza un kernel grafico per regolare i pesi assegnati in base alla visibilità dei segmenti vicini.

Queste due tecniche combinate consentono di ottenere ottimi risultati nella segmentazione di forme 3D, umanoidi e non.

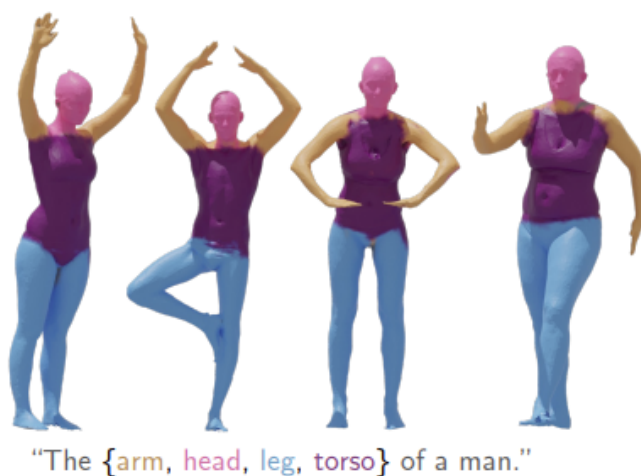


Figura 3.8: Esempio di segmentazione di una mesh umana con SATR.

L'utilizzo di tale progetto è stato poi abbandonato in favore di Blender, in cui sono state compiute operazioni di scripting che verranno poi illustrate nel dettaglio nel capitolo successivo di questa trattazione.

3.2.6 Dataset

Sebbene una descrizione dei dataset sia stata inserita per completezza nel capitolo precedente per descrivere il SOTA, in questo progetto di tesi non sono stati utilizzati. Le reti individuate e utilizzate hanno riportato in output risultati piuttosto performanti e mesh abbastanza accurate e dettagliate: per tali motivi infatti non

sono stati effettuati degli ulteriori training delle reti e non ci si è serviti dei dataset per l'integrazione.

Tuttavia, è stato utilizzato un dataset di immagini per testare la robustezza del sistema e verificare la generalizzazione delle reti in scenari non visti durante la fase di allenamento e sviluppo, **Celeb-FBI** [112].

Tale dataset è composto da immagini full-body e frontali alla fotocamera di celebrità, estratte da scene televisive e video pubblici. Include una varietà di soggetti in termini di età, etnia, espressioni facciali e condizioni di illuminazione, risultando utile per valutare la stabilità e la qualità delle ricostruzioni 3D generate dal sistema. Nonostante non sia stato impiegato per il training, questo dataset ha fornito una base realistica per validare le prestazioni delle reti utilizzate e osservare eventuali limiti in contesti più diversificati rispetto a quelli previsti dalle reti stesse.



Figura 3.9: Esempio di immagini presenti nel dataset Celeb-FBI.

3.3 Tecnologia utilizzata

3.3.1 Sistema Operativo

Ubuntu [109] è un sistema operativo distribuito da **Linux** nel 2004, basato sul ramo *unstable* di Debian. È composto prevalentemente da un software libero con licenza GNU GPL, ma include anche il supporto per software proprietario, offrendo un sistema operativo flessibile e accessibile. Progettato principalmente per computer desktop, Ubuntu presenta varianti dedicate a server, tablet, smartphone e dispositivi IoT, con particolare attenzione al supporto hardware.

La distribuzione di Ubuntu è gestita dall'azienda britannica Canonical Ltd., fondata da Mark Shuttleworth, che prevede rilasci con supporto a lungo termine (LTS) ogni due anni e versioni con supporto a breve termine ogni sei mesi.

Ubuntu è basato sull'ambiente desktop GNOME, mentre in passato utilizzava l'interfaccia Unity. È progettato per fornire un'interfaccia semplice e intuitiva, ma al tempo stesso completa e potente. Tra i punti di forza si trovano:

- facilità d'uso per utenti di qualsiasi livello.
- ottimo riconoscimento e supporto hardware.
- vasto catalogo di software, costantemente aggiornato.
- strumenti grafici di gestione intuitivi.

La versione desktop è pensata per soddisfare le esigenze di un utente medio, come navigare in Internet, gestire documenti e immagini, svago e comunicazione. Inoltre, offre un'ampia gamma di applicazioni libere e strumenti di amministrazione del sistema, gestibili sia singolarmente sia in modo centralizzato tramite l'Ubuntu Control Center, paragonabile al pannello di controllo di Windows o a strumenti come YaST di OpenSUSE.

Questo progetto di tesi è stato implementato con la **versione 22.04** di Ubuntu.

3.3.2 Python

Lo scripting e la programmazione sono stati implementati utilizzando il linguaggio **Python** [2].

Python è un linguaggio di programmazione moderno, gratuito e dalla sintassi semplice, per cui risulta molto versatile e adatto a svariati ambiti di applicazione, tra cui sviluppo di siti o di applicazioni web sia ad alto che a basso livello, realizzazione

di interfacce grafiche (*GUI*), amministrazione di sistema, calcolo scientifico e numerico e creazione di applicazioni, database, grafica 3D, videogiochi ecc. Il suo utilizzo presenta svariati vantaggi:

- linguaggio multi-paradigma, che supporta sia la programmazione procedurale (che fa uso delle funzioni) sia la programmazione ad oggetti sia diversi elementi della programmazione funzionale (come iteratori e generatori).
- portabile e utilizzabile su diverse piattaforme (Unix, Linux, Windows, iOS ecc) e integrabile con altri linguaggi di programmazione.
- possibilità di scaricare un gran numero di librerie nonché di modelli aggiuntivi creati e mantenuti dalla community.
- adotta un meccanismo di gestione automatica della memoria (garbage collection) che si occupa automaticamente dell'allocazione e del rilascio della memoria, consentendo al programmatore di usare liberamente le variabili.
- linguaggio di alto livello performante e intuitivo, che permette di compilare automaticamente i programmi in un formato chiamato bytecode prima di essere eseguiti, garantendo prestazioni efficaci.

L'utilizzo di Python è stato integrato con librerie di terze parti e framework per garantire il funzionamento delle reti neurali prese in considerazione. Tra queste:

- **Tensorflow**: libreria open source per l'apprendimento automatico, sviluppata da Google. Viene utilizzato per formare e distribuire modelli di machine learning, per classificazione delle immagini, per elaborazione del linguaggio naturale e l'analisi delle serie temporali, consentendo quindi di specificare l'architettura di un modello (numero di livelli, funzione di ottimizzazione, ottimizzazione ecc.).
- **Pytorch** [4]: framework open source (sviluppato da Facebook) per l'apprendimento automatico basato sul linguaggio di programmazione Python e sulla libreria Torch. Quest'ultima è una libreria utilizzata per creare reti neurali profonde, utilizzata dai data scientists soprattutto per la ricerca e per lo sviluppo delle applicazioni nel campo dell'intelligenza artificiale (AI). PyTorch è progettato per accelerare il passaggio tra prototipazione di ricerca e distribuzione.
- **OpenCV** (Open Source Computer Vision Library): libreria software open source sviluppata originariamente da Intel per computer vision e machine learning. OpenCV supporta un'ampia varietà di linguaggi di programmazione come Python, C++, Java, ecc. Può elaborare immagini e video per identificare oggetti, volti o persino la grafia di un essere umano. Questa libreria può essere combinata con Numpy per l'ottimizzazione delle operazioni numeriche.

- **Numpy**: libreria open source che fornisce supporto per array e matrici di grandi dimensioni e multidimensionali e che dispone di una raccolta di funzioni matematiche di alto livello per operare sugli array. Tra le caratteristiche principali di Numpy: supporta tecniche avanzate di slicing e di indicizzazione, riduce la necessità di cicli espliciti su array attraverso funzioni efficienti, si integra facilmente con librerie come Pandas, Matplotlib e Scipy, offre funzionalità di algebra lineare, trasformata di Fourier e numeri casuali.

Anche se non si tratta né di una libreria né di un framework bensì di una piattaforma di elaborazione in parallelo e di un modello di programmazione sviluppato da NVIDIA, è doveroso citare l'utilizzo di **CUDA**. Quest'ultima permette parallelizzare l'utilizzo delle GPU NVIDIA per accelerare diverse attività di calcolo che prevedono operazioni matematiche e dati complessi, dalle simulazioni al deep learning, all'elaborazione di immagini e segnali. A differenza della programmazione tradizionale sequenziale delle CPU, CUDA consente di suddividere le attività in sottoattività più piccole che possono essere eseguite simultaneamente sulla GPU, ottenendo notevoli incrementi di prestazioni e di velocità rispetto alle implementazioni tradizionali di CPU.

All'interno del progetto di tesi sono state utilizzate varie versioni di Python così come diverse versioni delle librerie sopracitate a seconda dei requisiti specifici delle singole reti.

3.3.3 Anaconda

Anaconda [6] è una distribuzione dei linguaggi di programmazione Python e R progettata specificamente per applicazioni nei campi dell'apprendimento automatico, delle data sciences e dell'intelligenza artificiale: il suo obiettivo è quello di semplificare la gestione e la distribuzione dei pacchetti. È disponibile per Linux, Windows e MacOS ed è stata sviluppata da Anaconda Inc. nel 2012.

I pacchetti, in Anaconda, vengono organizzati dal gestore dei pacchetti *conda*, utilizzato sia per Python che per altri linguaggi di programmazione; tramite *conda* è possibile installare oltre 7.500 pacchetti direttamente da PyPI (300 dei quali vengono già scaricati automaticamente all'interno dell'ambiente di sviluppo). Anaconda mette a disposizione anche una *GUI* (interfaccia grafica utente) chiamata Anaconda Navigator che offre una soluzione alternativa e più user-friendly all'interfaccia a riga di comando per la gestione dei pacchetti. È inoltre possibile accedere ad *Anaconda Public Repository*, che comprende un totale di circa 8000 pacchetti open source per applicazioni di data science e di machine learning.

Infine, Anaconda è uno strumento che permette di creare ambienti virtuali per isolare i vari progetti Python: tali ambienti sono utili per gestire dipendenze specifiche dei progetti senza interferire con altri progetti o con l'ambiente Python globale.

Per questo progetto di tesi è stata utilizzata una versione ridotta di Anaconda, chiamata **Miniconda** [6], che include esclusivamente Conda, Python e un piccolo numero di altri pacchetti essenziali. Ciò rende questa versione più leggera rispetto a quella integrale.

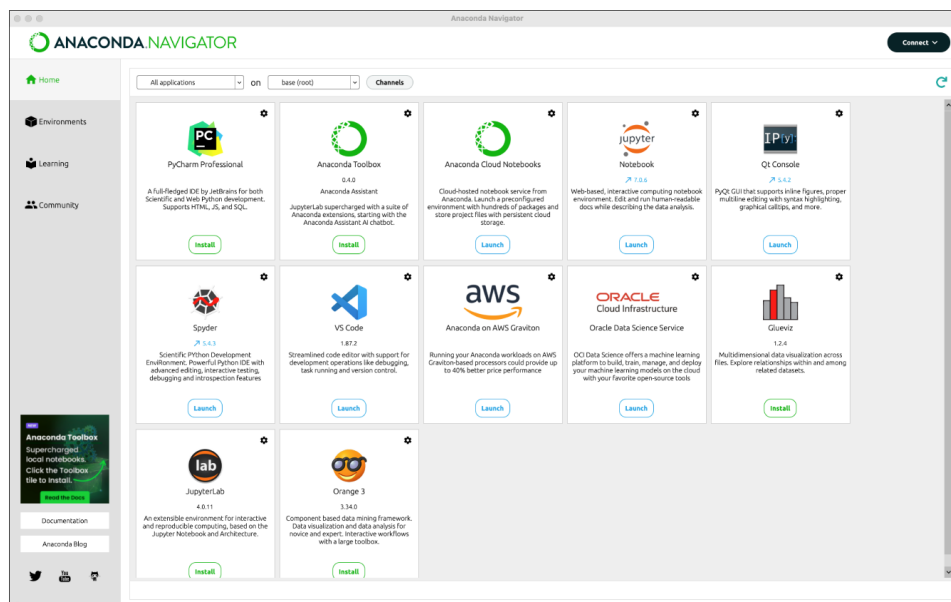


Figura 3.10: Interfaccia di Anaconda Navigator per l'installazione di pacchetti.

3.3.4 Pycharm

PyCharm [4] è un ambiente di sviluppo integrato (*IDE*) destinato alla programmazione nel linguaggio Python, sviluppato dalla Software House ceca JetBrains. PyCharm offre supporto integrato, oltre che per Python, anche per database SQL e NoSQL, Jupyter, Git, Conda, PyTorch, TensorFlow, Hugging Face, Django, Flask, FastAPI ecc.

Il software fornisce un'analisi del codice, nonché un editor intelligente per l'auto-completamento del codice e per suggerimenti in fase di scrittura, un debugger grafico, un collaudatore di unità (*unit tester*) integrato, l'integrazione con i sistemi di controllo di versione e il supporto allo sviluppo di applicazioni web. Nelle sue ultime versioni mette a disposizione anche un'assistente virtuale e con altre funzionalità, grazie all'integrazione con l'AI.

PyCharm è distribuito come multiplatforma e funziona su Microsoft Windows, macOS e Linux. È un ambiente di sviluppo che viene spesso utilizzato professionalmente, e per questo motivo JetBrains ne offre due versioni: una professional e una community gratuita e open-source. Per questo progetto di tesi è stata utiliz-

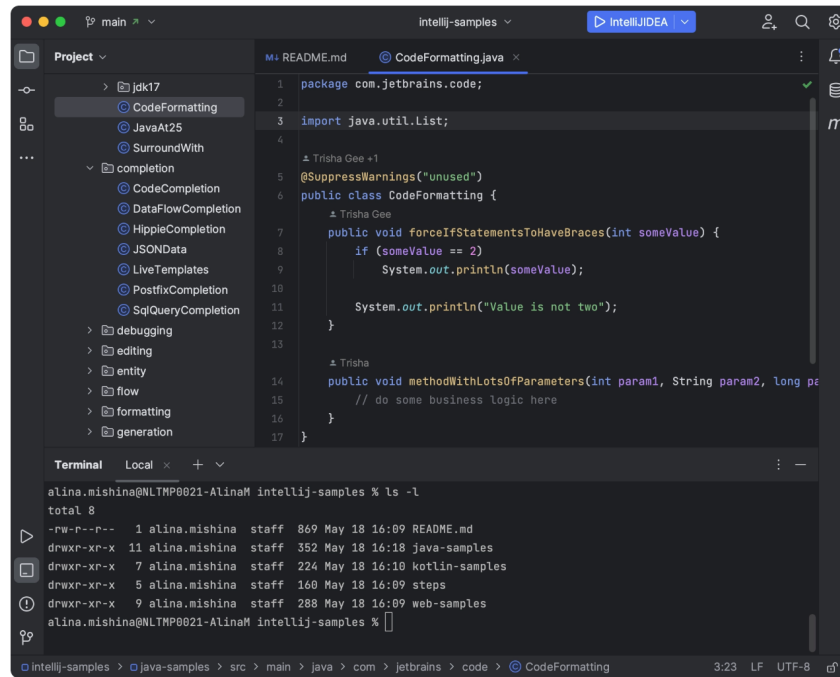


Figura 3.11: Interfaccia di Pycharm.

zata la **Community Edition**, precisamente la **versione 2024.2.3**, scaricabile e utilizzabile gratuitamente senza limiti di tempo.

3.3.5 Blender

Blender [1] è una piattaforma gratuita e open source in grado di gestire l'intero flusso di lavoro 3D, dalla modellazione, al rigging, all'animazione, al montaggio video, alla composizione, al rendering e al texturing di immagini tridimensionali e bidimensionali, alla creazione di videogiochi. Tra le sue funzionalità c'è anche la possibilità di utilizzare di raytracing e lo scripting in Python; quest'ultimo permette di personalizzare l'applicazione e sviluppare strumenti specializzati.

Per il gran numero di funzionalità di cui è dotato, Blender è paragonabile, per caratteristiche e complessità, ad altri noti programmi per la modellazione 3D (tra cui Maya, Cinema 4D, 3D studio Max ecc.) Blender è un software multiplatforma e funziona in modo ottimale su Linux, Windows e MacOS: la sua interfaccia utilizza OpenGL per garantire un'esperienza uniforme su tutti i sistemi. Essendo un progetto community-driven, distribuito sotto licenza GNU General Public License (GPL), Blender permette al pubblico di apportare modifiche al codice, contribuendo all'introduzione di nuove funzionalità, alla risoluzione rapida di bug e al miglioramento generale dell'usabilità.

Per questo progetto di tesi è stata utilizzata la **versione 4.2.2 LTS** di Blender ed è stato testato anche sulla versione **4.4.0**. All'interno del software sono stati importati i modelli creati dalle reti neurali e i modelli di vestiti. E' stata inoltre utilizzata l'API di Blender per il supporto dello scripting in Python al fine di drappare i vestiti sul corpo creato e di unire le varie mesh derivanti da diversi step della pipeline in un unico modello di avatar 3D.

Gli approfondimenti sono presenti nella sezione dedicata del Capitolo 4 di questo documento.

3.3.6 AI Background Remover

Si tratta di un tool online basato sull'intelligenza artificiale [8] per la rimozione automatica dello sfondo dalle immagini. Il servizio consente agli utenti di caricare una fotografia in input (in formato JPEG, PNG o WEBP), la quale viene elaborata attraverso modelli di deep learning specializzati nella segmentazione semantica. In pochi secondi, lo sfondo viene isolato e rimosso, mantenendo intatto il soggetto principale dell'immagine. L'output con sfondo trasparente può essere scaricato in formato PNG, per essere riutilizzata in contesti grafici, presentazioni, design o produzione multimediale.

Questo tipo di tecnologia si basa su reti neurali convoluzionali (CNN) addestrate su grandi dataset di immagini segmentate, e rappresenta un'applicazione per l'editing automatico delle immagini.

Va inoltre sottolineato che online esistono numerosi strumenti simili, molti dei quali offrono funzionalità analoghe, spesso gratuite, per la rimozione dello sfondo. Tra i più noti si possono citare *remove.bg*, *Adobe Express Background Remover* e *Canva Background Remover*, ciascuno con caratteristiche, algoritmi e qualità di output differenti.

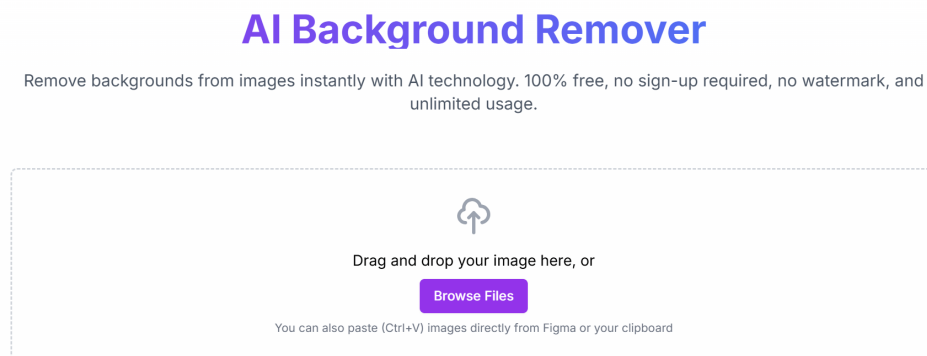


Figura 3.12: Interfaccia del tool AI Background Remover.

Capitolo 4

Pipeline di lavoro

Dopo aver approfondito come avviene la ricostruzione di mesh tridimensionali a seconda dei vari approcci adottati, e aver descritto le tipologie dei metodi di modellazione che costituiscono il SOTA e le tecnologie utilizzate in questo progetto di tesi, nel quarto capitolo ci si concentrerà sulla pipeline di lavoro. Verranno infatti descritti tutti gli step necessari per la ricostruzione di un avatar umanoide 3D a partire da una singola immagine RGB monoculare come input.

4.1 Pipeline

Gli step che devono essere compiuti all'interno della pipeline di lavoro del nostro progetto sono i seguenti:

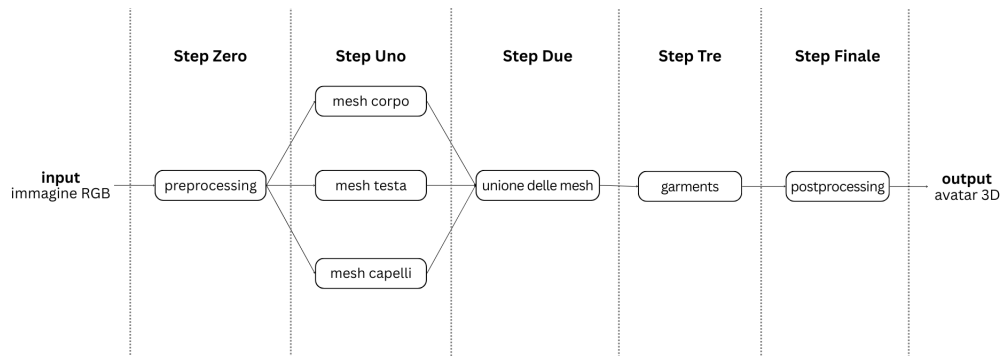


Figura 4.1: Pipeline tematica del progetto.

- Inizializzazione, ovvero la preparazione dei dati per le reti.
- Ricostruzione del corpo utilizzando la rete LVD.
- Ricostruzione dei vestiti, implementato in Blender.

- Ricostruzione della testa con la rete DECA.
- Ricostruzione dei capelli, utilizzando la rete HairStep.
- Segmentazione della mesh e modellazione, sviluppato in Blender.

Dunque, ricevendo come input una singola immagine RGB e svolgendo tutti in passaggi appena elencati, si avrà in output il modello dell'avatar che rappresenta nello spazio tridimensionale il soggetto raffigurato nell'immagine 2D iniziale.

Per ottimizzare le tempistiche del processo, alcuni step di cui si compone il metodo possono essere parallelizzati; una rappresentazione più efficace è quella proposta nell'immagine sottostante.

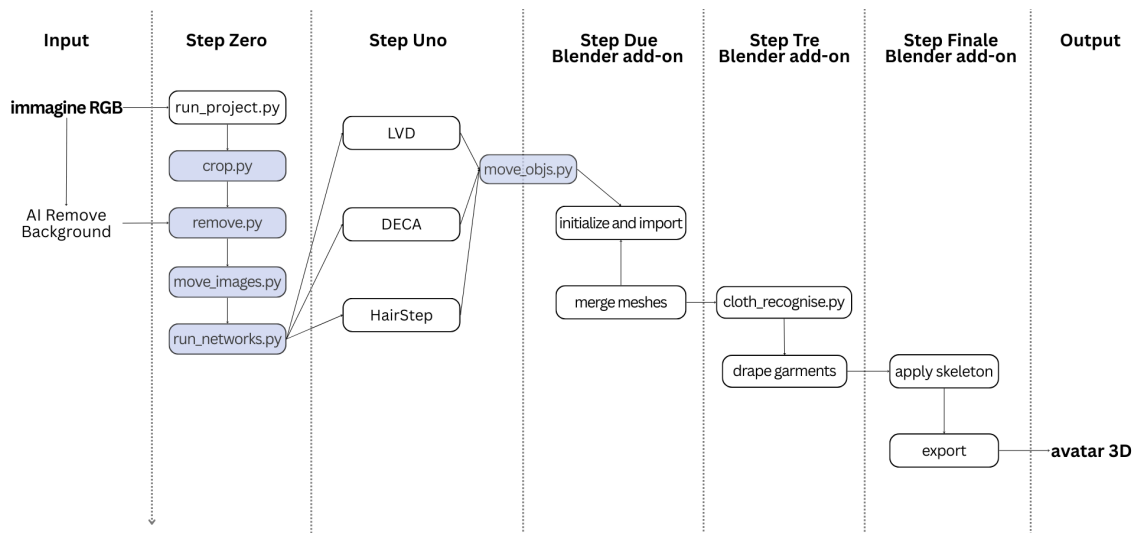


Figura 4.2: Pipeline dettagliata del progetto.

Come si può vedere nell'immagine, la fase di Inizializzazione rappresenta lo step Zero della pipeline, dal momento che si occupa di compiere operazioni sull'immagine in input, creando nuove immagini che verranno utilizzate negli step successivi a seconda dei requisiti specifici di ogni rete. Al termine dello Zero, si potranno avviare parallelamente la ricostruzione del corpo, quella del viso e quella dei capelli, dal momento che tali processi sono indipendenti gli uni dagli altri e costituiscono lo step Uno. Nel momento in cui quest'ultimo si conclude, potrà iniziare lo step Due, in Blender, ovvero l'unione di tutte le mesh create nel passaggio precedente: dovrà essere sostituita la testa del modello creato dalla rete LVD con quella ricevuta in output da DECA, su cui verranno posizionati i capelli di HairStep. Seguirà poi la fase in cui avverrà la ricostruzione dei vestiti, sempre in Blender, e il loro draping sul corpo dell'avatar, che costituiranno lo step Tre della nostra pipeline. Infine, ci sarà la possibilità di applicare lo scheletro al body per renderlo animabile e di esportare il modello così creato e riggato.

4.2 Input

L'input al progetto sarà, come già specificato, una singola immagine 2D RGB monoculare che potrà essere scattata con qualsiasi tipo di dispositivo elettronico. Per raggiungere risultati il più accurati possibile, l'input deve avere le seguenti specifiche:

- la persona ritratta deve essere a figura intera e frontale alla fotocamera. Particolare attenzione va dedicata al posizionamento delle testa, che non dovrà essere ruotata o inclinata;
- sarebbe preferibile se la testa del soggetto coincidesse con il bordo superiore dell'immagine e i piedi con quello inferiore, occupando tutto lo spazio disponibile in verticale;
- il soggetto dovrà essere ritratto con le braccia completamente distese e in un angolo che comprende la loro posizione lungo i fianchi fino all'altezza delle spalle (T-pose)
- è preferibile che lo spazio che fa da background al soggetto non sia troppo "disturbante", per consentire una rimozione dello sfondo più precisa;
- la luce dell'ambiente in cui si trova il soggetto nel momento in cui viene scattata la fotografia non deve creare ombre eccessivamente marcate né sul soggetto stesso, né sullo sfondo.

La risoluzione dell'immagine, invece, sarà a discrezione dell'utente e influente nei passaggi successivi della pipeline di lavoro, così come verranno trascurati i parametri e il posizionamento della fotocamera.

Nelle prove che sono state effettuate nella fase di test del progetto si sono utilizzate immagini estratte dal dataset FBI a risoluzione variabile.



Figura 4.3: Esempio di immagini utilizzate come input.

4.3 Step Zero

Lo step Zero, l’inizializzazione, è costituito da un progetto composto da sei script, scritti ad hoc per gli scopi di questa tesi, che permettono di automatizzare il pre-processing degli input, il funzionamento delle reti neurali e l’estrazione degli output dai vari progetti.

Per semplicità, tutti gli script del progetto verranno descritti in questa sottosezione della tesi; tuttavia alcuni di questi verranno eseguiti negli step successivi della pipeline. L’immagine [4.2] aiuta a comprendere meglio i vari momenti in cui vengono runnati.

4.3.1 Crop.py

Questo script, data l’immagine originale in input al progetto, ne fa dapprima il *crop* ed eventualmente il *resize* per creare nuove immagini aderenti alle specifiche delle reti utilizzate. In particolare, si avranno due immagini in output (una 224x224 per DECA e una 512x512 per HairStep). A tal proposito è necessario individuare dapprima i *landmark bidimensionali* dell’intero corpo: basandosi sulle posizioni di questi, infatti, ne deriva l’altezza a cui l’immagine originale deve essere tagliata. Si è fatto uso della libreria **MediaPipe Pose Landmarker**, che consente di rilevare i landmarks dei corpi umani in un’immagine o video. Questo strumento può essere utilizzato per identificare le posizioni chiave del corpo, analizzarne la postura e categorizzare i movimenti, basandosi su modelli di machine learning (ML) che funzionano sia con immagini singole che con video. L’output consiste nei landmarks della posa del corpo in coordinate bidimensionali dell’immagine e in coordinate tridimensionali nel mondo reale.

Per entrambe le immagini, è sufficiente effettuare il crop all’altezza della linea che collega i punti chiave corrispondenti alla spalla destra e sinistra del soggetto (landmarks 23 e 24). Si deriva dapprima l’input di DECA (224x224) che verrà salvato e successivamente ingrandito per arrivare a una dimensione di 512x512, necessaria per HairStep.

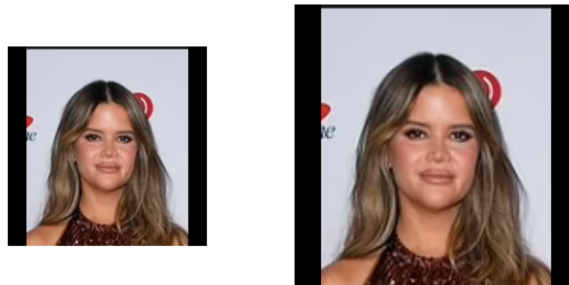


Figura 4.4: Immagini croppate per la rete DECA (a sinistra) e HairStep (a destra).

4.3.2 Remove.py

Il secondo script a cui vengono passate le immagini appena create è `remove.py`, che si occupa della creazione delle due immagini necessarie a far funzionare LVD. Da specifiche, tale rete richiede infatti due immagini: una in cui viene rappresentato il soggetto su uno sfondo nero uniforme, e un'altra, simile alla prima, in cui il soggetto però viene coperto da una maschera di colore bianco, mantenendo sempre il background nero. Per ottenere questi risultati si è inizialmente pensato di utilizzare un approccio completamente algoritmico attraverso la libreria **rembg**, che utilizza una rete neurale per eseguire la rimozione dello sfondo. L'algoritmo è addestrato per identificare e separare gli oggetti in primo piano da quelli in secondo piano, producendo immagini con background trasparenti (e quindi dotate di un canale alpha per rappresentare la trasparenza di tali parti). Tuttavia, nonostante questa libreria costituisca un'automatizzazione di un processo, ha molta difficoltà nell'estrarre il soggetto se sono presenti delle ombre particolarmente marcate sullo sfondo (in questo caso, infatti, le ombre vengono processate e integrate come se fossero parte del soggetto stesso) o se lo sfondo è particolarmente disturbante. Nella seconda immagine creata, inoltre, il soggetto bianco sul background nero ha contorni non ben definiti e abbastanza sfumati, cosa che trae in inganno la rete LVD nel momento in cui deve definire le dimensioni reali del corpo da ricostruire.



Figura 4.5: Immagini cui è stato rimosso il background con il tool AI Background Remover.

Per ovviare a tale problematica si è deciso di utilizzare un tool online (**AI Background Remover**), che compie le stesse operazioni della libreria *rembg* ma giunge a risultati più puliti e precisi. L'immagine in input al progetto verrà quindi dapprima caricata nell'interfaccia del tool, le verrà rimosso lo sfondo e verrà salvata in formato .png per essere fornita come input allo step Zero, insieme a quella originale

scattata dall'utente.

Nello script `remove.py`, verrà poi correttamente scalata per rispettare le specifiche della rete LVD (che richiede una risoluzione di 256x256 in input). Verranno qui create le due immagini per la rete, applicando uno sfondo nero uniforme al posto di quello trasparente e andando a ricoprire il soggetto con una maschera di colore bianco, sempre su sfondo nero.

Questo script avrà quindi in output due immagini, necessarie alla rete LVD.



Figura 4.6: Immagini in input alla rete LVD.

4.3.3 `Move_images.py`

Il terzo script si occupa di copiare le immagini create nei due step precedenti e di posizionarle correttamente nelle cartelle dei progetti che verranno utilizzati nello step Uno. In questo modo, le reti avranno già l'immagine in input all'interno della propria cartella di progetto, a cui potranno accedere agevolmente e senza l'intervento dell'utente.

4.3.4 `Run_networks.py`

Questo script racchiude i comandi utilizzati per far partire le elaborazioni effettuate dalle tre reti neurali di cui si compone lo step successivo. Dopo il preprocessing delle immagini e dopo averle copiate nelle rispettive cartelle dei progetti, questo script attiva, per ogni rete considerata, il suo ambiente virtuale ed esegue i comandi da linea di comando previsti per il funzionamento della rete stessa. Per come è organizzato il codice, i processi avverranno in serie e non in parallelo dal momento, se eseguiti contemporaneamente, potrebbero andare a saturare le risorse del calcolatore utilizzato.

4.3.5 Move_objs.py

A questo punto viene implementata una logica simile a quella di `move_images.py`, ma svolgendo la funzione opposta. `move_objs.py`, infatti, si occupa di recuperare i file prodotti in output dalle reti neurali, rinominarli e copiarli all'interno di una cartella specifica (`networks_output`). I file di tipo `.obj` o `.ply` verranno poi importati nel progetto Blender, mentre per quanto riguarda gli altri (quelli di tipo `.png` o `.json`) verranno utilizzati per definire il colore e la texture delle varie parti della mesh.

4.3.6 Run_project.py

Infine, l'ultimo script (in ordine logico di trattazione e non temporale di esecuzione) di cui si compone il progetto è `run_project.py`: questo codice permette di lanciare l'esecuzione in serie dei cinque script appena descritti e quindi, indirettamente, anche il funzionamento delle reti neurali. Pertanto, dovrà essere eseguito come primo e unico script, dal momento che, per come è stata implementata la logica della pipeline, non ci sarà necessità da parte dell'utente di eseguire comandi ed operazioni manualmente.

4.4 Step Uno

Come già accennato, in questo step si avranno tre operazioni in parallelo, svolte da altrettante reti neurali (ricostruzione della mesh del corpo, del viso e dei capelli). Tali passaggi possono essere svolti contemporaneamente poichè sono indipendenti gli uni dagli altri. Dal momento, però, che si tratta del funzionamento di tre progetti in parallelo, bisogna fare attenzione alle prestazioni del PC su cui viene lanciato il codice e alle limitazioni della GPU e della sua scheda grafica, per evitare che il processo si blocchi e provochi un crash della macchina.

Andiamo ora ad analizzare i vari punti nello specifico. Di seguito i vari step seguiranno una numerazione progressiva: l'ordine non è temporale, ma è stato scelto coerentemente con quello utilizzato in tutto l'elaborato di tesi.

4.4.1 Step 1.1

Si occupa della ricostruzione del corpo effettuata dalla rete LVD.

Come già messo in luce nel capitolo precedente, tale rete non si concentra tanto sulla ricostruzione della posa quanto piuttosto sulla modellazione di una mesh umana che sia il più possibile aderente nella sua shape a quella ricevuta in input. A tal proposito, vengono utilizzati 300 parametri per definire la forma del corpo, conformemente con l'ultima versione rilasciata nel 2020 da SMPL e che coincide con

quella utilizzata per SMPL-X (prima di allora, il modello base SMPL ne utilizzava solamente 72). Il corpo in output segue, quindi, quello del modello SMPL per numero di vertici e di facce triangolari, e si compone di 24 giunti (come si può vedere nell'immagine sottostante): per ogni giunto viene determinata la rotazione lungo gli assi x , y , z . Tali valori di rotazione, tuttavia, non sono assoluti ma sono relativi a quelli del giunto appena precedente nella catena cinematica. Per accertarsi che il soggetto venga correttamente ricostruito in T-pose, si va a modificare direttamente il codice della mesh, imponendo a zero tutte le rotazioni dei giunti per avere una mesh con le braccia perfettamente perpendicolari al busto. Tale semplificazione è fondamentale nello step di modellazione dei vestiti, di cui si parlerà diffusamente in seguito.

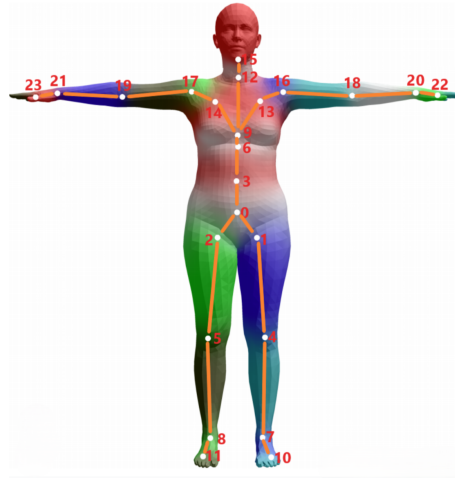


Figura 4.7: Visualizzazione dei giunti del body SMPL.

La rete LVD riesce a ricostruire più accuratamente la forma del corpo rispetto agli altri metodi SOTA, andando ad analizzare il corpo nudo del soggetto presente al di sotto dello strato dei vestiti; in questo modo si riesce ad avere una mesh “pura” in cui non viene incorporata la geometria e lo spessore dei capi di abbigliamento presenti sulla sua superficie. Non vi è quindi necessità, come invece accade per altre reti, che i vestiti del soggetto siano il più possibile aderenti per evitare che la rete interpreti in modo errato la geometria del body se vestito con abiti troppo larghi che ne nascondono eccessivamente la silhouette.

Da requisiti, inoltre, viene richiesto che il soggetto nell'immagine scattata in input sia a figura intera e che occupi verticalmente tutto lo spazio della fotografia, con i piedi in corrispondenza del bordo inferiore e la testa vicina a quello superiore. Attraverso tale richiesta si cerca di risolvere il problema dell'altezza della mesh del body costruita da LVD: se il soggetto raffigurato nell'immagine in input è troppo distante dalla fotocamera la mesh rappresentata viene eccessivamente scalata verticalmente e ne viene ridotta l'altezza. Inoltre, solo in alcuni casi la rete riesce ad identificare

la porzione dell'immagine in corrispondenza dei capelli del soggetto: ne deriva che in certi casi la mesh finale sia dell'altezza corretta (se questi ultimi vengono esclusi dalla modellazione), mentre in altri risulti leggermente più alta (se la chioma viene considerata parte della mesh del body).

Un ulteriore punto da mettere in luce è il fatto che le mesh modellate, seppur ricalchino accuratamente la shape dell'input, appartengono al genere neutro. Ciò è piuttosto influente nel caso in cui si abbia il corpo di un uomo in input ma risulta piuttosto limitante nel momento in cui bisogna rappresentare una donna, poiché la mesh manca degli aspetti che caratterizzano solitamente la silhouette femminile. A tal proposito è stata modificata direttamente la rete: per ogni ricostruzione che dovrà essere effettuata verrà specificato se si tratta di un uomo o di una donna. A seconda dell'input ricevuto dall'utente, LVD andrà a selezionare il modello corretto (`SMPL_FEMALE.py` per i corpi femminili o `SMPL_MALE.py` per quelli maschili) per la ricostruzione. In questo modo, le mesh create saranno genderizzate e le loro silhouette saranno più adatte al sesso cui appartengono. Bisogna però precisare che la rete è stata allenata con il modello neutro, mentre la miglioria che è stata apportata interviene solo nella fase di test: i caratteri maschili e femminili verranno quindi solamente accennati nella mesh finale.

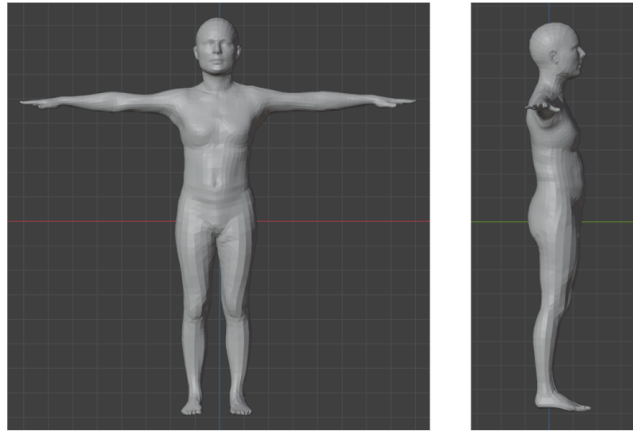


Figura 4.8: Ricostruzione LVD di un corpo femminile.

La mesh del body è ricavata runnando il codice dello script `test.LVD.images.py` e dando in input alla rete due immagini in formato 256x256, la prima in cui viene rappresentato il soggetto e la seconda in cui è presente la sua maschera di segmentazione, entrambe con background nero e con soggetto posizionato al centro. Queste immagini derivano dallo step Zero, con l'esecuzione di `remove.py`. L'input numero uno viene letto e convertito in formato RGB e viene isolato dallo sfondo attraverso la maschera binaria dell'input numero due; viene successivamente normalizzata secondo le convenzioni di **ImageNet** (ampia base di dati di immagini utilizzata in

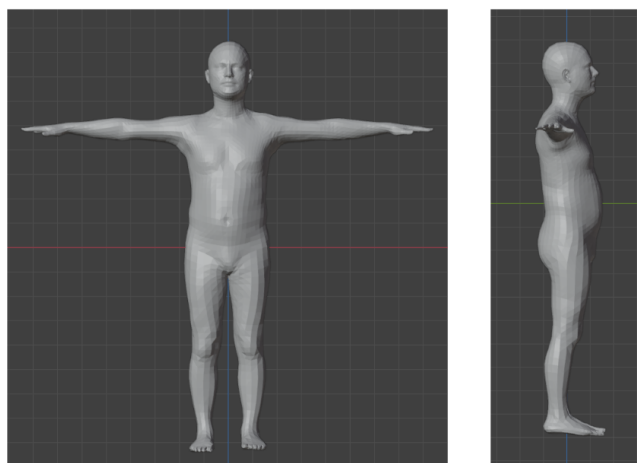


Figura 4.9: Ricostruzione LVD di un corpo maschile.

ambiti di VR, AI ecc.) per garantire coerenza con il modello di deep learning. Dopo il preprocessing, si ha lo step di inferenza del modello, in cui la rete neurale prende in input l'immagine elaborata e genera una prima stima della mesh tridimensionale del corpo: si tratta di un processo iterativo, articolato in cinque aggiornamenti successivi per migliorare l'accuratezza della ricostruzione. In questa fase vengono stimati 6890 punti nello spazio tridimensionale che rappresentano la superficie del corpo umano. Per rendere la ricostruzione il più fedele possibile alla realtà, è presente inoltre una fase di ottimizzazione basata su SMPL: ci si serve a questo punto di una rete neurale addestrata a generare pose corporee realistiche regola i parametri del modello per minimizzare la differenza tra la mesh stimata e il modello SMPL standard. Tale processo avviene attraverso 500 iterazioni di discesa del gradiente, con l'obiettivo di ridurre l'errore di allineamento e garantire che la forma e la postura risultanti siano coerenti con le caratteristiche del soggetto. Inoltre, viene applicato un vincolo basato su una distribuzione statistica delle pose umane per evitare configurazioni fisicamente irrealistiche.

All'interno dello script viene imposto che la mesh abbia origine al centro della sua geometria e sia posizionata all'origine degli assi. Il body, inoltre, poggia i piedi su un piano perpendicolare all'asse z e quindi parallelo al piano individuato dagli assi xy : il modello sarà quindi orientato lungo l'asse z . Tutte le mesh create negli altri step della pipeline saranno conformi a questo orientamento.

4.4.2 Step 1.2

Si occupa della ricostruzione della mesh della testa attraverso l'utilizzo della rete DECA.

Nonostante un modello di testa sia già presente per la mesh del corpo nudo creato nello step 1.1, LVD non risulta essere molto accurata nella modellazione dei dettagli facciali come può invece esserlo una rete che si occupa solo di questo task. A tal proposito è stato selezionato un progetto che non ricreasse soltanto il viso ma tutta la testa, in modo da andare a sostituire l'intero capo già presente sul corpo con quello creato in questo step, più dettagliato e aderente all'input (questo passaggio verrà effettuato in un momento successivo tramite script direttamente in Blender). La rete inoltre modella il solo scalpo senza i capelli: aspetto assolutamente positivo dal momento che anche in questo caso l'acconciatura del soggetto verrà ricreata nello step 1.4 da una rete specializzata in tali mesh.

La rete riceve in input una sola immagine 224x224 opportunamente ricavata grazie allo script `crop.py` allo step Zero della pipeline, e da cui vengono generate la posa della testa, la forma, la geometria dettagliata del volto e le informazioni sull'illuminazione. Per ricavare il modello tridimensionale della testa e i files ad esso correlati viene utilizzata la versione demo della rete (script `demo_reconstruct.py` all'interno della cartella `demos` del progetto).

L'esecuzione del codice è gestita tramite argomenti e parametri scritti direttamente nella linea di comando, dando la possibilità all'utente di specificare il percorso delle immagini, la cartella in cui salvare gli output, i tipi di files da generare e altre opzioni, come la scelta del dispositivo di esecuzione (CPU o GPU). Il codice è ottimizzato per funzionare su GPU con CUDA, riducendo i tempi di elaborazione. Grazie alla sua flessibilità e robustezza, il sistema è in grado di gestire immagini acquisite in condizioni non vincolate, adattandosi a variazioni di illuminazione, pose e occlusioni del volto.

Tale script utilizza svariate librerie, tra cui le già citate OpenCV per la gestione delle immagini, NumPy per le operazioni numeriche, Torch per l'elaborazione con reti neurali e, infine, SciPy per la memorizzazione dei dati. Successivamente, vengono importati i moduli di DECA, tra cui il modello della rete allenata in fase di training, i dataset del progetto per il confronto con l'immagine in input, le funzioni di utilità per il processamento delle immagini e le impostazioni proprie della rete. Dopo aver definito la cartella di output, il codice carica l'input, che nel nostro caso è un'immagine singola ma può anche essere costituito da più fotografie o addirittura da un video. Durante il caricamento, ci si serve di un **face detector** che elabora l'immagine in input, individuando e ritagliando l'area del volto in modo da centrarla correttamente. Dopo aver preprocessato l'input, il modello della rete DECA viene configurato e caricato: a questo punto possono essere attivate diverse opzioni specificate nella linea di comando nel momento in cui viene runnato il codice, come per esempio l'uso di un modello di texture e la scelta del tipo di rasterizzatore per la visualizzazione finale. Per ogni immagine in input, lo script esegue una codifica della geometria facciale servendosi di un encoder, che permette di convertire i pixel della fotografia in una rappresentazione numerica dei parametri tridimensionali del

volto. Tale rappresentazione viene poi passata ad un decoder che genera la mesh 3D, i landmarks facciali nello spazio tridimensionale e la texture finale del volto. Successivamente, lo script salva i risultati nella cartella di destinazione. I file esportati possono includere mappe di profondità, punti chiave 2D e 3D, modelli tridimensionali in formato .obj, file .mat per l'analisi in MATLAB, e immagini visualizzate con le diverse modalità di rendering. Viene inoltre salvato un file in formato .json in cui è presente il codice RGB della carnagione del soggetto. Dal momento che sul viso sono presenti 68 landmarks facciali e che la maggior parte di loro coincide con elementi come le pupille, l'arcata sopraccigliare, le bocca ecc. e quindi non è posizionata direttamente sulla pelle del soggetto, si va ad estrarre il colore in corrispondenza di quelli numero 27, 28, 29 e 30 (che coincidono con il naso) e dove il colore è quello corretto della carnagione del volto. Tale codice RGB verrà poi importato in Blender e utilizzato come materiale per la mesh del body.

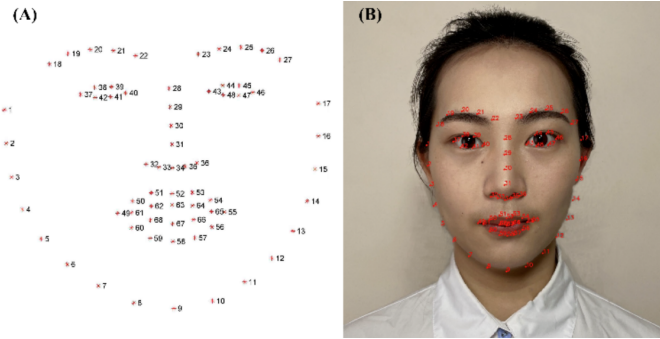


Figura 4.10: Visualizzazione dei landmarks del viso.

La mesh ricevuta in output può inoltre essere animata, deformando realisticamente le rughe del viso e utilizzando una seconda immagine da cui viene estrapolata l'espressione facciale che è trasferita sulla mesh creata a partire dalla prima immagine. Tale funzionalità, tuttavia, va oltre gli scopi previsti in questo elaborato di tesi.

Lo script di DECA, nel momento in cui salva la mesh della testa, impone che la sua origine sia al centro della sua geometria e che la rotazione sia coerente con quella imposta al modello del corpo, per semplificare l'utilizzo della mesh in Blender. L'output .obj della testa verrà infatti successivamente importato in Blender e posizionato sul body del soggetto.

Un fattore da tenere in considerazione: la rete non ricostruisce direttamente sulla mesh in output la geometria di “elementi accessori” come per esempio i *facial hairs*, gli occhiali, i piercing ecc. Tali oggetti verranno tuttavia ricavati dall'immagine originale nel momento in cui si va ad estrarre la mappa della texture del volto, in cui compaiono anche gli elementi accessori. Come si può infatti vedere nell'immagine sottostante, che rappresenta un soggetto con la barba, i *facial hairs* non vengono

modellati direttamente sulla geometria della testa, ma vengono comunque rappresentati andando ad applicare la texture sulla mesh. Questo approccio permette quindi di ricostruire tutti gli “elementi” che si trovano sul viso in maniera facile e veloce (avendo inoltre la possibilità di ricreare anche dettagli non strettamente materiali, come per esempio il make up del soggetto, le cicatrici, i tatuaggi ecc. che comunque non verrebbero modellati sulla geometria del volto in quanto si trovano direttamente sulla superficie della pelle).

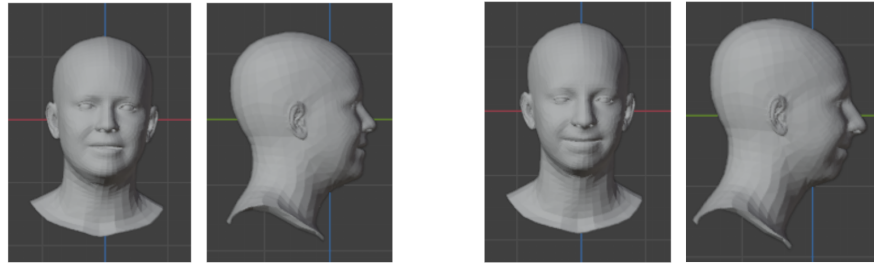


Figura 4.11: Mesh di teste create con la rete DECA.

Infine, l’immagine della texture del viso verrà successivamente utilizzata in Blender ma prima verrà modificata all’interno di DECA. Tale immagine si compone infatti dei dettagli e del colore delle sole parti del volto, mentre per la restante parte del capo assume una tinta grigia, il cui codice RGB è (178,178,178). Questo colore viene sostituito all’interno del progetto con quello estratto in corrispondenza dei landmarks del naso, che quindi rappresenta la carnagione del soggetto e che viene utilizzato per la texture del body: l’immagine finale risulta quindi correttamente texturizzata e pronta per essere applicata sulla mesh della testa. Tale approccio tuttavia non porta a risultati ottimali nel caso in cui il volto del soggetto non abbia una tinta omogenea e/o sia sottoposto a luci e ombre piuttosto marcate. A tal proposito il colore della viso subisce un blending graduale con il colore del corpo (dal momento che si tratta di un approccio automatizzato può portare ad artefatti e a textures non particolarmente precise nella zona di transizione, problema che verrebbe ovviato andando a modificare l’immagine manualmente).



Figura 4.12: Textures del viso derivanti dalla rete DECA.

4.4.3 Step 1.3

In questo step della pipeline avviene la ricostruzione dei capelli, effettuata per mezzo di HairStep.

Tale rete riceve una singola immagine in input (in formato 512x512) in cui è rappresentato il volto del soggetto e la sua chioma. Questo progetto, tuttavia, rappresenta una limitazione: le mesh dei capelli vengono modellate in modo accurato e aderente alla realtà solo nella parte frontale, mentre in quella posteriore alla testa e sui suoi lati la mesh potrebbe assumere delle forme geometriche particolari e non realistiche. Ciò è dovuto al fatto che la rete riceve solo un'immagine in input in cui il soggetto è posto frontalmente alla fotocamera, per cui il retro della testa risulta, giustamente, impossibile da prevedere da parte di HairStep. Per esempio, nel caso di acconciature con capelli raccolti, e quindi invisibili frontalmente nell'immagine target, la rete ricrea i capelli corti dal momento che l'unica informazione che ha sulla chioma è la vista sullo scalpo.

L'input per la rete deriva dallo script `crop.py`. La rete funziona eseguendo quattro script diversi, che si trovano nella cartella scripts del progetto.

Nel primo avviene il preprocessing dell'immagine in input, che viene segmentata per isolare il volto e i capelli: utilizzando il modello **Segment Anything (SAM)** si generano maschere binarie in corrispondenza del viso e dei capelli per distinguerli dal resto dell'immagine. Successivamente, viene generata una mappa delle ciocche per mezzo di una rete neurale U-Net (addestrata specificatamente per l'analisi della struttura capillare) per rappresentare la direzione e la distribuzione dei capelli. Viene inoltre individuata la profondità del volto, essenziale per conferire tridimensionalità alla chioma, andando ad analizzare la struttura dell'immagine e assegnando ad ogni pixel un valore di profondità: la depth map viene poi normalizzata e visualizzata utilizzando una scala cromatica con colori diversi in corrispondenza della diversa profondità. Il secondo script invece si occupa dell'estrazione della posa e dei landmarks facciali utilizzando **3DDFA_V2**, un modello di deep learning per la stima della geometria del viso in 3D. L'obiettivo è quello di individuare l'orientamento della testa: a tal proposito utilizza **FaceBoxes**, un algoritmo per il rilevamento del volto, che restituisce in output l'angolo di inclinazione del capo e la sua posizione nello spazio.

Il terzo script, invece, cerca di ottimizzare la proiezione della fotocamera per allineare landmarks estratti in precedenza con la vista prospettica dell'immagine di input: l'obiettivo è quello di regolare i parametri della fotocamera in modo che i landmarks proiettati corrispondano alla posizione effettiva del volto nell'immagine. L'ottimizzazione viene eseguita attraverso un processo iterativo basato su discesa del gradiente: la funzione di perdita valuta la discrepanza tra i punti chiave stimati e quelli reali, regolando iterativamente i parametri della fotocamera per minimizzare questa differenza. Dopo più di duecento iterazioni, l'immagine ottimizzata con i

landmarks proiettati viene salvata, insieme ai parametri finali della fotocamera: la finalità è quella di garantire che la ricostruzione 3D sia accurata e coerente con la vista prospettica dell'immagine originale.

L'immagine target preprocessata e gli output ricavati fino a questo momento vengono caricati nella fase finale della ricostruzione, ovvero nel quarto script della demo. Una volta preparati i dati di input, lo script carica e inizializza due reti neurali: **OccNet**, responsabile della geometria complessiva del modello 3D, definendo la forma del viso e del cuoio capelluto, e **OrienNet**, dedicata alla generazione della struttura tridimensionale dei capelli, che sfrutta la mappa delle ciocche per determinare l'orientamento e la distribuzione dei singoli.

La chioma viene infine salvata ed esportata sia in formato .obj che in formato .ply, in cui le ciocche di capelli vengono rappresentate come singoli filamenti. La texture della chioma inoltre non viene determinata in maniera esplicita dalla rete, per cui è necessario estrarla in un modo alternativo, andando a modificare lo script originale di HairSep. Precisamente, verranno combinati gli input con la maschera di segmentazione dei capelli al fine di isolare questi ultimi all'interno dell'immagine originale; a questo punto si estrarrà il colore dominante in formato RGB, che salvato in un file .json.



Figura 4.13: Chiome in formato .ply ottenute da HairStep.

4.5 Step Due

A questo punto si dispone di tutti gli elementi necessari per lo step Due, ovvero quello in cui vengono uniti i risultati ricevuti in output dalle reti nello step precedente, al fine di creare la mesh completa dell'avatar. Si dispone, quindi, di un corpo, completo e senza vestiti, a cui bisogna sostituire la testa e applicare i capelli sullo scalpo di quest'ultima.

L'esecuzione di questo step della pipeline avverrà interamente in Blender per mezzo di un **add-on**, opportunamente creato per gli scopi di questa tesi.

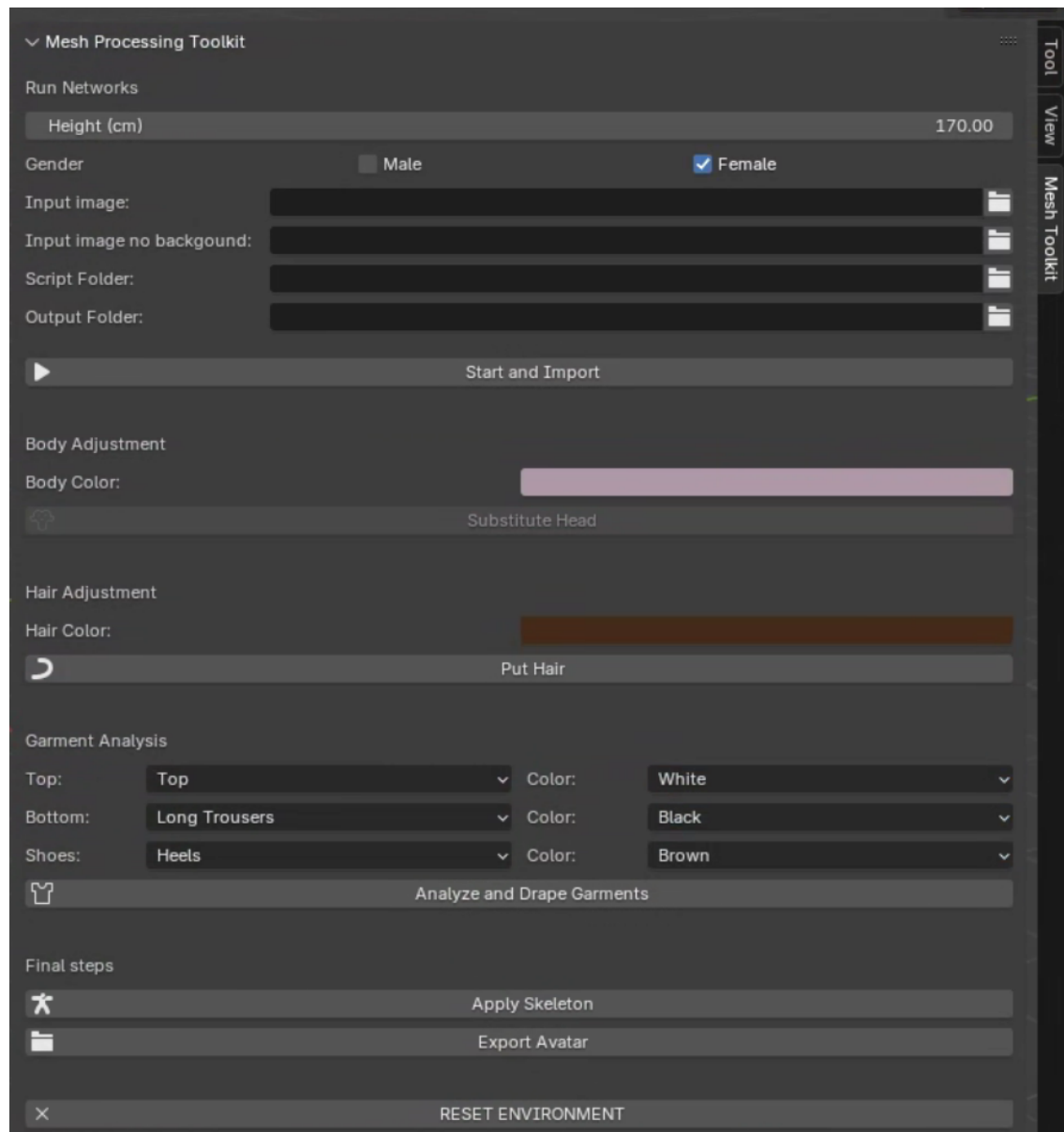


Figura 4.14: Mesh toolkit add-on.

Tale add-on, in formato .zip al cui interno si trovano una prima cartella contenente tutti gli script utilizzati, una seconda in cui sono presenti i capi di abbigliamento utili nel progetto e il file .py dell'addon vero e proprio, dovrà essere importato all'interno dell'ambiente del progetto, mediante il tab Edit - Preferences nella sua interfaccia. Dopo averlo installato da disco, la UI dell'add-on sarà visibile nella sidebar dell'interfaccia di Blender. Accedendo al suo tab, si avranno a disposizione vari pulsanti, ognuno collegato ad uno script che si occupa di svolgere una determinata funzione. A un primo sguardo si possono inoltre notare uno spazio per un input testuale, due checkbox, due campi gradiente e sei menù a tendina. Si procederà ora con ordine nella descrizione di ciascuno di questi elementi e delle loro funzionalità.

Il primo bottone è **Start and Import**, che esegue tre funzioni principali. Per prima cosa runna lo script `run_project.py`, di cui si è parlato allo step Zero. Premendo il pulsante, infatti, si avvieranno progressivamente tutti i passaggi collegati a tale script, e verranno eseguite le elaborazioni da parte delle tre reti neurali. Prima di utilizzare tale bottone, è necessario specificare l'altezza del soggetto (in centimetri, da scrivere nell'apposito campo della UI), il suo sesso, il path del progetto Crop (indicato come input folder) e il path della cartella in cui salvare gli output. Al termine delle elaborazioni, i risultati in output verranno salvati nella cartella specificata dall'utente (`networks_output` di default).

Successivamente ci si occupa di importare tutte le mesh all'interno del progetto. Verranno infatti caricati i file di tipo .obj e .ply che sono stati restituiti in output dalle reti neurali (nello specifico, il corpo di LVD, la testa di DECA e i capelli creati di HairStep). Di questi, solo la prima mesh rimarrà visibile per essere utilizzata nel passaggio successivo. Verrà inoltre importata un'armatura umanoide (creata grazie all'add-on *Rigify* e in formato .fbx) opportunamente modificata in modo che le braccia siano perpendicolari al busto e che lo scheletro si trovi quindi perfettamente in T-pose, in modo da essere correttamente allineato con la posa del corpo (sono state effettuate inoltre modifiche per quanto riguarda l'allineamento delle ossa dei piedi e l'aggiunta di un osso, *ReferenceBone*, utilizzato successivamente per il posizionamento dei capelli).

Verranno letti i valori presenti nei file .json in output da DECA e da HairStep, in cui sono presenti i colori RGB rispettivamente del body e dei capelli. Tali tinte verranno applicate come valori default nei relativi campi della UI (che tuttavia possono essere modificati a discrezione dell'utente). Blender utilizza uno spazio colore lineare per i calcoli interni legati all'illuminazione e al rendering, in modo da garantire risultati fisicamente accurati. Tuttavia, per visualizzare i colori nell'interfaccia utente, applica una correzione gamma che converte i valori lineari in sRGB, lo standard usato dalla maggior parte dei monitor. Questo processo rende i colori visivamente più brillanti rispetto ai valori originali inseriti, anche se il colore applicato alla mesh

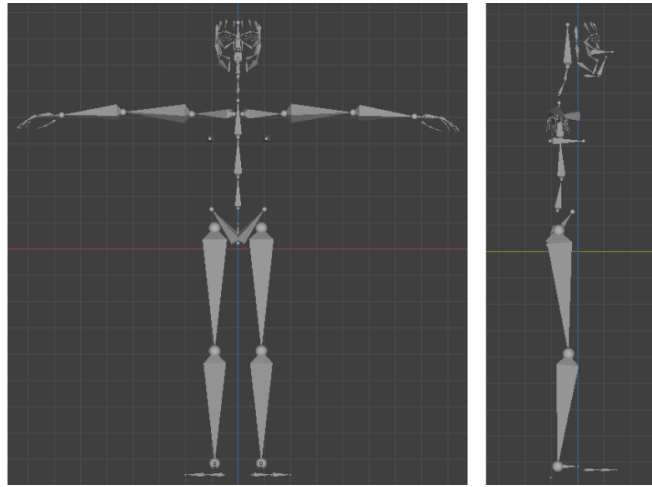


Figura 4.15: Scheletro adattato alla T-pose.

resta corretto a livello fisico.

La seconda cartella da importare nel progetto racchiude tutte le mesh di vestiti, in formato .obj, che, in questa prima fase, verranno nascoste nella viewport.

Il passaggio successivo si occupa di scalare, traslare e ruotare l'amatura in modo da adattarla perfettamente, soprattutto in altezza, alla mesh del body.

Dapprima il corpo verrà scalato in tutte le direzioni, al fine di essere dell'altezza specificata dall'utente all'interno della UI dell'add-on. Il body verrà poi suddiviso in due parti attraverso un piano parallelo a yz: per ognuna delle due semi mesh create, si andrà a determinare il punto più basso in corrispondenza dei piedi del modello. Alla semi-mesh che avrà una coordinata z più bassa verrà applicato un *Mirror modifier* per specchiarla rispetto all'asse z. La mesh che si andrà a creare sarà dunque perfettamente simmetrica al fine di risolvere eventuali e possibili problemi di modellazione da parte della rete LVD (come una diversa lunghezza delle gambe o delle braccia nelle due parti).

Verranno infine effettuate le operazioni necessarie per adattare le dimensioni dell'amatura a quelle del corpo. In questa prima fase, lo scopo dello scheletro non è tanto quello di essere riggiato sul body per andare a modificarne la posa, bensì quello di costituire un riferimento sull'asse z per il posizionamento in altezza della testa, dei capelli e dei vari capi di abbigliamento, e di calcolare le dimensioni del body in questi punti precisi per evitare intersezioni tra mesh (il rig in questa fase verrà nascosto).

Successivamente, come previsto dalla pipeline, si dovrebbe premere il pulsante **Substitute Head**, che permette di sostituire in quattro passaggi la testa originale della mesh LVD con quella ricostruita dalla rete DECA.

Tali passaggi verranno spiegati per completezza, sebbene nel progetto tale pulsante

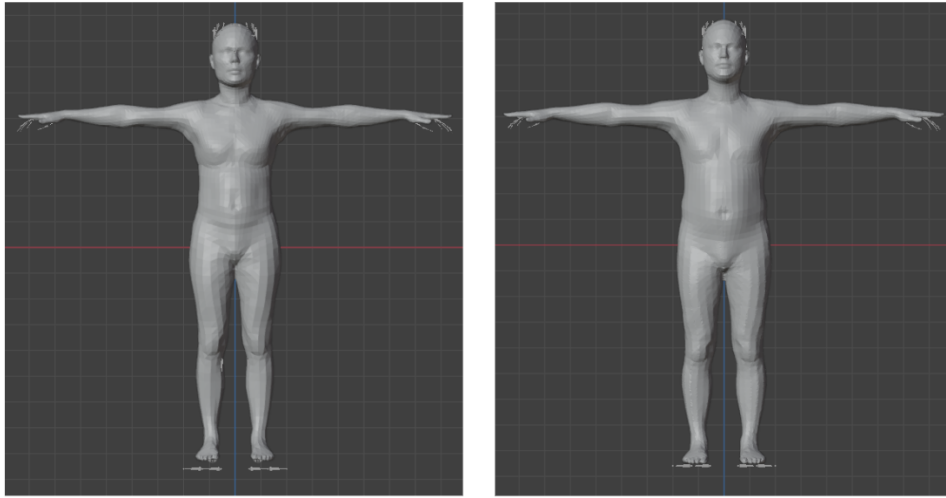


Figura 4.16: Corpi LVD a cui è stato applicato lo scheletro.

risultati disattivato. Tale decisione è stata presa in quanto l'algoritmo di sostituzione non consente, allo stato attuale, di ottenere in maniera completamente automatizzata una fusione pulita e "neutra" tra le due mesh. Nonostante siano stati effettuati numerosi tentativi con approcci differenti tramite scripting in Blender, i risultati presentavano artefatti geometrici visibili e buchi nella mesh difficili da correggere unicamente via codice, richiedendo invece un intervento manuale.

Per ovviare a questo problema, si è quindi scelto di mantenere la testa originale generata dalla rete LVD, seppur meno dettagliata e precisa rispetto a quella ottenuta da DECA. Tale scelta consente di preservare la coerenza geometrica complessiva della mesh ed evitare problematiche nel calcolo delle metriche di valutazione.

La procedura di sostituzione della testa, così come progettata, si articola in quattro passaggi principali:

- **Allineamento della testa:** la mesh della testa generata da DECA viene orientata in modo da risultare perfettamente frontale e coerente con l'asse del corpo della mesh LVD. In questa fase si controllano e si rifiniscono le rotazioni sui tre assi per garantire l'allineamento geometrico.
- **Scalamento della testa:** per evitare sproporzioni (come colli eccessivamente lunghi o teste troppo grandi/piccole), si effettua uno scaling della testa DECA. Il parametro di riferimento adottato è la distanza tra la punta del mento e la punta del naso, che viene calcolata sia sulla mesh LVD che su quella DECA, consentendo una scala coerente tra le due teste.
- **Taglio delle mesh:** si identificano due coppie di punti sia sulla mesh LVD che su quella DECA: uno nella parte anteriore, appena sotto il mento (dove inizia effettivamente il collo), e uno nella parte posteriore alla stessa altezza lungo

l'asse Z (all'altezza dello scalpo). Questi punti definiscono i piani di taglio per eliminare la testa originale LVD dal resto del corpo e la parte inferiore della mesh di DECA, al fine di isolare le porzioni da unire. Alla prima mesh (il body) verrà attribuito un nuovo materiale, la cui tinta corrisponde a quella presente nella UI dell'add-on del campo specifico; alla seconda invece verrà applicata la texture estratta dalla rispettiva rete e importata nel progetto.

- **Fusione delle mesh:** una volta eseguito il taglio, si procede all'unione delle due sezioni mediante l'applicazione di alcuni modificatori in Blender, primo fra tutti lo *Shrinkwrap*, per aderire la mesh DECA al collo LVD nel modo più fluido possibile. Successivamente si applicano i modificatori *Subdivide* e *Smooth* per raffinare la transizione tra le due superfici e migliorarne la continuità visiva.

Sebbene questa procedura, in teoria, mantenga proporzioni coerenti e allineamenti corretti, nella pratica la linea di giunzione tra le due mesh rimane spesso troppo evidente. In alcuni casi, inoltre, la mesh finale presenta anomalie topologiche, come buchi o distorsioni difficilmente gestibili senza un intervento manuale, compromettendo la qualità del risultato finale.

La testa DECA verrà quindi tenuta separata dal body.

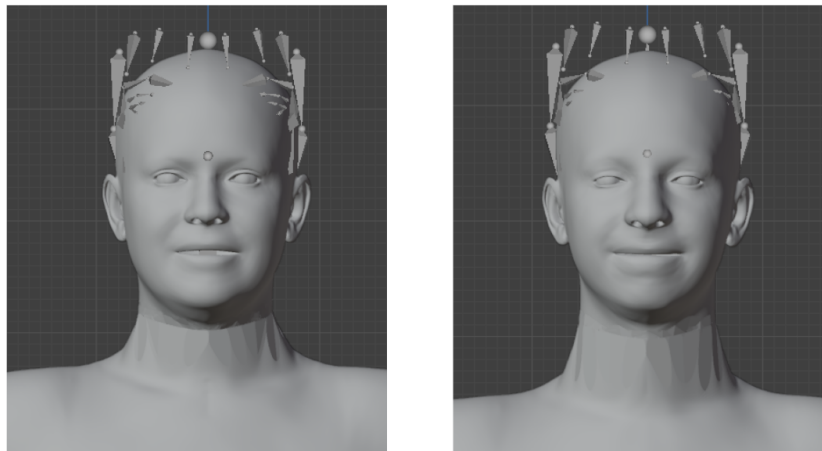


Figura 4.17: Sostituzione della testa DECA sul corpo LVD.

A questo punto mancano i capelli per completare la mesh, che verranno importati in Blender nel loro formato .ply e verranno allineati all'rig del corpo.

Per posizionare correttamente i capelli, lo script, attivato dal tasto **Put hair on Body**, cerca la posizione del punto più alto del capo e ne calcola le coordinate nel mondo 3D. A questo punto, si passa all'analisi della mesh dei capelli per determinare quale sia il vertice più alto che si trova sulla superficie interna della mesh, quindi nella parte più vicina al centro della testa. La chioma verrà dunque traslata sull'asse z per essere opportunamente posizionata sullo scalpo, mentre sull'asse delle x e delle

y la sua posizione rimarrà al momento invariata. Vengono successivamente determinate le massime dimensioni della testa e dei capelli su questi assi, e il rapporto tra queste grandezze determinerà quanto deve essere scalata la mesh dei capelli in queste direzioni. Si svolgeranno poi delle ulteriori operazioni per fare in modo che nessuna parte del capo rimanga scoperta e che, allo stesso tempo, la chioma non vada a coprire troppo il viso (a tal proposito, viene utilizzato un "reference bone", parallelo all'asse y, che aiuta a determinare quanto in profondità deve essere posizionata la chioma).

Al termine delle operazioni di posizionamento e traslazione, la mesh dei capelli verrà converita in una curva, a cui si applicherà un *solidify modifier* per aggiungere spessore ai filamenti dei capelli. In questo modo l'avatar sarà anche completo di chioma, a cui verrà attribuito un nuovo materiale il cui colore è quello inserito nel rispettivo campo input della UI dell'add-on, similmente a come è avvenuto per il body.



Figura 4.18: Posizionamento dei capelli sul corpo LVD.

4.6 Step Tre

In questo step, il penultimo di cui si compone la pipeline, ci si concentra completamente sulla ricostruzione dei vestiti. Tale processo è implementato interamente in Blender, attraverso script ad hoc, e necessita della mesh del corpo completa modellata allo step Due. A partire dall'interfaccia dell'add-on, l'utente andrà ora a selezionare, attraverso tre diversi menù a tendina (uno per la parte superiore del corpo, uno per la parte inferiore e uno per le scarpe, con i rispettivi menù a tendina

con la scelta del colore da attribuire al vestito selezionato), i capi di abbigliamento che preferisce per vestire il soggetto tra quelli disponibili nel progetto Blender. Nello specifico, nel progetto sono state previste le seguenti categorie di indumenti:

- per la parte superiore del corpo si hanno una T-shirt, una maglia dalle maniche lunghe, una canotta, un vestito corto e una giacca (con il suo relativo top);
- per quella inferiore sono stati modellati i pantaloni di tre lunghezze diverse (shorts, bermuda e pantaloni lunghi) e tre gonne (anche in questo caso mini, al ginocchio e lunga);
- per le scarpe invece si hanno un paio di anfibi, un paio di ciabatte, un paio di scarpe da ginnastica e uno di ballerine da donna con un leggero tacco.

Per quanto riguarda le textures da applicare agli indumenti, invece, le tinte previste da progetto sono le seguenti: cream, light gray, black, white, light blue, brown, beige, red, blue, green, yellow, pink, purple, gray, orange.

Al termine della selezione del vestiario da parte dell'utente, un apposito bottone, **Analyze and Drape Garments**, permetterà di runnare, nell'ordine, gli script `cloth_recognise.py` e `drape_garments.py`. Al termine della sua esecuzione verranno resi visibili nella viewport di Blender solo i capi di abbigliamento selezionati con il loro relativo colore, modellati ad hoc sul corpo del soggetto (operazione di cui si parlerà in questa sottosezione). Si tratta quindi di un approccio completamente algoritmico in cui non vengono utilizzate le reti neurali.

Come già anticipato brevemente nel capitolo 3 di questo elaborato, può essere effettuata una suddivisione di vestiti, oltre che per categoria di indumento, anche per come è costruita la loro mesh e per come avviene l'applicazione della stessa sul body.

La prima tipologia è costituita da mesh modellate a mano, che si compongono solo della faccia anteriore e posteriore dell'indumento che vogliono rappresentare, separate tra loro nello spazio e tenute insieme solo grazie alle linee di cucitura: i vestiti vengono così decomposti nei loro sew pattern. Nello specifico, appartengono a questo primo genere tutte le categorie di abbigliamento ad eccezione delle scarpe.

La loro pipeline di modellazione sul body del soggetto si articola in tre passaggi. Nel primo step, ci si occupa dello scalamento e della traslazione dell'indumento: si tratta di un passaggio piuttosto delicato poiché ci si deve assicurare che il vestito non interpenetri la mesh del corpo e che si adatti in maniera ottimale alla superficie di quest'ultimo. Avendo come riferimento di altezza le coordinate delle varie ossa dello scheletro, ad ogni garment viene dapprima attribuito un primo scalamento sull'asse delle z, specifico per ogni categoria. Per fare un esempio pratico, nel caso del vestito corto da donna si impone che la sua lunghezza vada dal punto più basso del bone

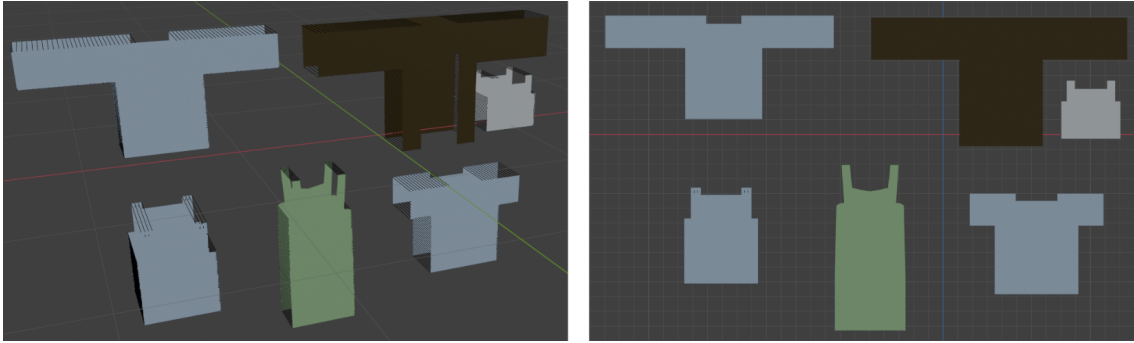


Figura 4.19: Modelli di vestiti per la parte superiore del body.

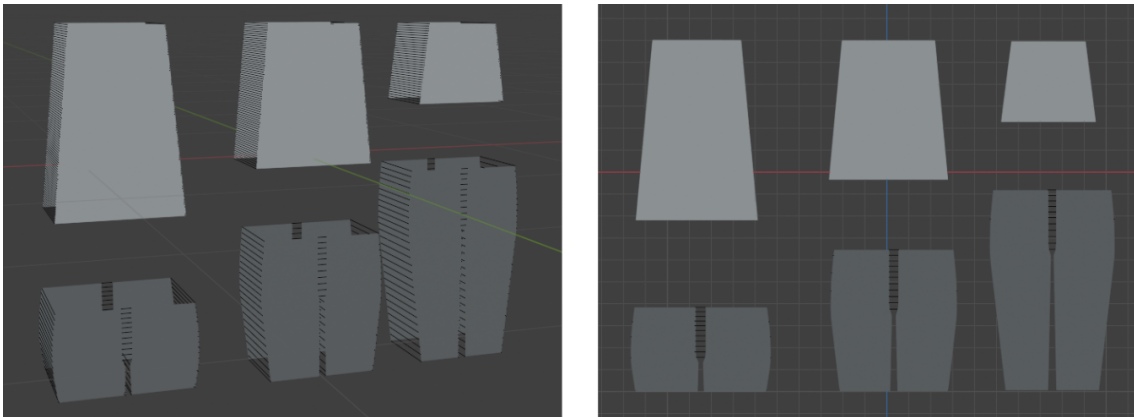


Figura 4.20: Modelli di vestiti per la parte inferiore del body.

spine.005 al punto medio dell'osso thigh.L. All'interno del range appena individuato si calcola la massima larghezza del corpo sull'asse delle x e delle y, ottenendo così un garment opportunamente scalato nelle tre direzioni. Per assicurarsi ulteriormente che non vi siano interpenetrazioni tra le due mesh, si aggiunge un looseness factor, che scala ulteriormente i vestiti di un piccolo valore che varia a seconda di casi (per il mini dress si parla di un fattore pari a 1.25). Successivamente, dopo aver collocato gli indumenti adattati alle dimensioni del body nella giusta posizione, mantenendo sempre l'origine degli stessi al centro della loro geometria, si passa al secondo step.

Ci si occupa, a questo punto, della *Cloth Simulation*: attraverso una seconda tipologia di codice (anche questa volta si ha uno script per ogni indumento) vengono definiti i parametri ottimali da settare per la simulazione. In particolare, al garment viene aggiunta la fisica del *Cloth*, ovvero del tessuto, i cui parametri assumono valori diversi a seconda della tipologia dei capi presa in considerazione, in modo impostare correttamente la cloth simulation. nello specifico si andranno a modificare i seguenti:

- **Quality steps:** determina la qualità della simulazione (a valori più alti corrispondono risultati più precisi, ma il processo è più lento). Settato a 15 per tutti i capi.
- **Stiffiness:** rappresenta la rigidità strutturale
 - Tension: determina quanto il tessuto resiste allo stiramento: un valore alto simula tessuti rigidi come il denim.
 - Compression: determina la resistenza alla compressione (per esempio, quanto facilmente si accartoccia).
 - Shear: controlla la resistenza alla distorsione diagonale del tessuto (utile per tessuti come la seta e materiali leggeri).
 - Bending: controlla la flessibilità del tessuto (più è basso, più il tessuto è morbido).
- **Damping,** ovvero lo smorzamento della simulazione, che riduce le oscillazioni e il rimbalzo del tessuto sul collider.
 - Tension: smorza lo stiramento del tessuto.
 - Compression: smorza l'effetto di schiacciamento.
 - Shear: smorza le torsioni diagonali del cloth.
 - Bending: riduce la flessibilità del tessuto.
- **Shape** (forma e deformazioni)
 - Sewing: viene abilitata la cucitura delle due parti di cui si compone la mesh, e in alcuni casi viene modificata anche la Sewing Force (per esempio nel caso delle tshirt e delle maglia a maniche lunghe).

- Shrink Factor: simula il restringimento o l'espansione del tessuto.

- **Cloth Collisions**

- Object Collision: attiva le collisioni con altri oggetti (attivata per tutte le tipologie).
- Distance: distanza minima tra il tessuto e altri oggetti per evitare compenetrazioni (settata al suo valore minimo di 0.001).
- Self-Collision: abilita la collisione con sé stesso (attivata per tutte le tipologie).
- Friction: resistenza del tessuto al contatto con altri oggetti (più è alto, più il tessuto aderisce alla mesh del collider): di default settata a zero, ma viene incrementata solo per certe tipologie, ovvero tutte quelle che vanno a ricoprire la parte superiore del corpo.
- Distance: distanza minima tra le stesse parti del tessuto per evitare interpenetrazioni (settata al suo valore minimo di 0.001).
- Quality: determina la qualità della simulazione per quanto riguarda le collisioni (il suo valore varia da un minimo di 8 ad un max di 10).

I parametri non presenti in questo elenco non verranno modificati, ma verranno lasciati al loro valore di default: tra questi la massa del cloth, che rimarrà settata a 0.3 kg, e le altre forze esterne, come il vento, la resistenza dell'aria ecc. con valore pari a 1. Altri aspetti della simulazione non verranno invece attivati: è il caso delle internal springs (le forze interne al tessuto) e della pressure. Non viene inoltre definito nessun tipo di vertici (ad esempio i pin group, vertici che rimangono fissi per evitare che il tessuto scivoli completamente dal body).

Durante la cloth simulation, il vestito andrà a “scontrarsi” con la mesh del body, a cui viene aggiunta la fisica del tipo *Collision*, su cui verrà drappato il tessuto. Anche in questo caso verranno definiti dei parametri, che variano a seconda del capo di abbigliamento preso in considerazione. Dal momento che si tratta di una cloth simulation si andranno a modificare solo i seguenti:

- **Damping Factor:** controlla quanto il tessuto viene rimbalzato quando colpisce il collider. Valori più alti smorzano il movimento, facendo aderire meglio il tessuto.
- **Thickness Outer:** determina lo spessore dell'esterno dell'oggetto collider. Un valore maggiore impedisce al cloth di avvicinarsi troppo alla mesh.
- **Thickness Inner:** definisce lo spessore interno del collider (utile per soft body ma meno rilevante per Cloth).

- **Friction:** determina quanto il tessuto scivola sul collider. A valori più alti corrisponde più attrito e il tessuto si attacca maggiormente.

Viene inoltre impostato un certo numero di frames che definiscono la durata della simulazione fisica. Per ogni indumento si determinerà poi uno specifico frame in cui il draping risulta maggiormente realistico e soddisfacente nella sua deformazione: in questo istante infatti verrà bloccata la simulazione, verrà applicato il *modifier Cloth* e la mesh iniziale del vestito manterrà la forma drappata sul corpo nello specifico frame. A questo punto si passa all'ultimo passaggio della pipeline, quello di refinement della mesh. In questo step verranno svolte diverse operazioni sui modelli di garment appena deformati al fine di renderli ancora più aderenti alla realtà e gradevoli alla vista. Nel caso in cui, al termine della simulazione, si vedessero ancora le linee di cucitura che uniscono le due parti della mesh si utilizzerà un *Merge by Distance* per rimuovere i vertici duplicati e unirli in uno solo, definendo un valore di threshold (soglia di fusione). Un valore troppo alto per questo parametro può eliminare dettagli importanti, mentre nel caso contrario potrebbe non unire abbastanza punti. Successivamente, la mesh verrà suddivisa per incrementarne il numero di vertici e verranno applicati sulla sua superficie due modificatori. Dapprima si utilizzerà, un *modifier Solidify* necessario per aggiungere spessore ad una mesh bidimensionale nello spazio 3D, convertendo una superficie sottile in un oggetto con volume. Lo spessore aggiunto sarà minimo, con offset tale da non interferire con la mesh del corpo sottostante. Successivamente verrà applicato uno *Smooth modifier*, per ammorbidire la geometria della mesh senza modificarne la topologia, andando a smussare gli spigoli e rendendo il modello più morbido: a seconda dei casi verrà incrementato o diminuito il parametro repeat (ovvero il numero di volte in cui l'effetto viene applicato sulla superficie della mesh).

Al termine di tutte queste elaborazioni si otterrà una mesh realistica per i vari indumenti, deformata rispetto a quella iniziale e modellata ad hoc sul corpo del soggetto.

La seconda tipologia di vestiti presente nel progetto è costituita dalle mesh che vengono importate già modellate; i garments di questa categoria si limitano ai modelli di scarpe (ovvero sneakers, anfibi, ciabatte, ballerine eleganti da donna) le cui mesh sono state scaricate da BlenderKit. Per fare in modo che calzino correttamente, ci si serve di un unico step. In questo script vengono svolte numerose funzioni: prima fra tutte, la mesh della calzatura viene preprocessata all'interno dell'ambiente di Blender, andando a definirne la giusta posizione e scalatura in base alle dimensioni del corpo in corrispondenza del piede (simile a come avviene per la tipologia uno, andando dunque a calcolare le altezze a cui collocare le scarpe in base alla coordinata z delle ossa del piede e scalandole nelle tre direzioni in base ai punti più larghi dello stesso).

A differenza di ciò che accade per la prima tipologia di vestiti, in cui l'indumento viene cucito direttamente sulla forma del body sottostante, qui la mesh è già

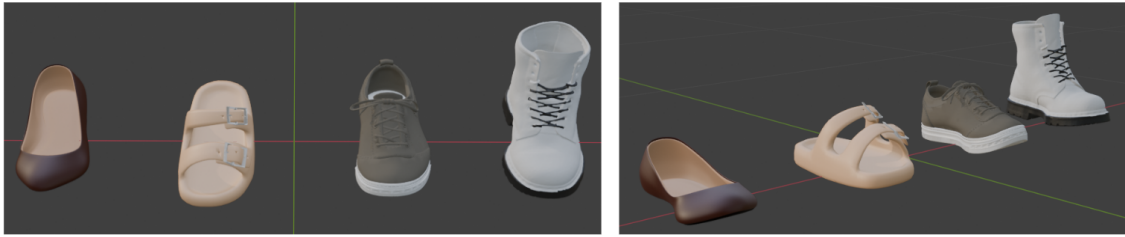


Figura 4.21: Modelli di scarpe presenti nel progetto.

modellata e definita per cui il rischio di interpenetrazioni tra questa e il corpo sottostante è decisamente maggiore. Nonostante venga utilizzato un *looseness factor* per aumentarne ulteriormente le dimensioni, talvolta non basta: parti della geometria delle scarpe potrebbero scomparire completamente all'interno di quella del body, andando a lasciare delle parti del piede scoperte. Per ovviare a tale errore, verrà applicato alla calzatura uno *Shrinkwrap modifier*, che serve per proiettarne la mesh sulla superficie di un'altra (il body), facendo aderire la prima alla forma della seconda. In tutti i casi considerati il wrap mode del modificatore (che specifica il metodo da utilizzare per determinare il punto più vicino sulla superficie dell'oggetto target per ciascun vertice dell'oggetto da modificare) verrà settato sull'opzione *Nearest Surface Point*. Per il parametro *Snap mode*, utile per controllare come il vertice viene spostato verso lo specifico punto target, invece verrà selezionata la modalità "Outside": in questo modo l'offset, ovvero la distanza che dovrà essere mantenuta tra le due mesh, sarà proiettata verso l'esterno del corpo, in modo che il capo di abbigliamento non ne compenetri la mesh e che gli venga conferito un piccolo spessore. Dopo essersi accertati che non ci siano interferenze reciproche, si procede ad applicare uno *Smooth modifier* sulla scarpa, per renderne la superficie gradevole alla vista e uniforme nella sua geometria. Questo script costituisce una pipeline piuttosto efficace per garantire una modellazione il più realistica possibile: tuttavia risultati migliori si otterrebbero sicuramente andando a modificare manualmente la mesh delle calzature dopo averle applicate sul body (opzione che però va oltre ad un approccio algoritmico automatizzato come quello che si vuole implementare all'interno di questo elaborato di tesi).

I passaggi appena descritti vengono effettuati solo sulla mesh della scarpa sinistra. Al termine del preprocessing del modello, verrà applicato un *mirror modifier*, che si occupa di specchiare la scarpa lungo l'asse z e rispetto alla mesh corpo. Questo risulta essere un metodo veloce ed efficace per posizionare le scarpe sul body, anche tenendo in considerazione il fatto che di solito le mesh delle calzature sono dotate di una geometria piuttosto complessa e ricca di vertici: lavorare quindi su una sola di queste permette di raggiungere i risultati desiderati in modo più veloce e meno oneroso dal punto di vista computazionale.

Al termine dell'esecuzione dello script si avrà quindi un corpo completo in tutte le sue parti e a cui sono stati applicati correttamente i vestiti e le scarpe selezionate.

4.7 Step Finale

A questo punto l'utente potrà accedere ad altri due bottoni presenti nell'add-on. Con il pulsante **Apply Skeleton**, infatti, gli verrà data la possibilità di applicare l'armatura sull'avatar finale, in modo da renderlo animabile.

Il bottone **Export Avatar** gli permetterà, infine, di esportare la mesh dell'avatar, riggata e completa di vestiti, in formato .fbx.

Reset Environment, poi, elimina tutte le mesh presenti nel progetto e svuota tutti i campi di input dell'add-on, riportandoli al loro valore di default.



Figura 4.22: Alcuni esempi di avatar completi.

Capitolo 5

Risultati ottenuti

5.1 Output

Il risultato finale sarà quindi un avatar 3D umanoide full body e completo di vestiti, a cui non è stato possibile sostituire efficacemente la testa originale con quella più aderente all'immagine in input ottenuta attraverso la rete DECA, ed i capelli. Tutte le parti della mesh saranno inoltre complete delle rispettive textures, permettendo di conservare le informazioni visive originali (i colori della pelle, dei vestiti e dei capelli), essenziali per una resa visiva realistica.

A partire da una singola immagine RGB in input, quindi, il sistema sviluppato è in grado di generare una mesh tridimensionale che unisce componenti provenienti da diverse reti. Il risultato ottenuto è un modello 3D unificato, pronto per essere utilizzato da software come Blender e compatibile con le pipeline di animazione o di rigging.

L'output finale può essere esportato in formati standard come `.obj` o `.fbx`, rendendolo utilizzabile in contesti applicativi differenti; le diverse componenti della pipeline di modellazione e della mesh stessa, inoltre, sono modulari, facilitando eventuali modifiche o personalizzazioni del modello.

Visivamente, l'avatar generato risulta fedele alle proporzioni e all'aspetto dell'individuo rappresentato nell'immagine di input, compatibilmente con i limiti dei modelli utilizzati e delle condizioni di acquisizione dell'immagine.

5.2 Metriche di valutazione

Nel contesto della ricostruzione tridimensionale e della generazione di avatar, le metriche di valutazione costituiscono strumenti efficaci per valutare in modo oggettivo o soggettivo la qualità degli output prodotti. Tali metriche possono essere di natura

numerica o visiva e sono utilizzate per quantificare in modo sistematico la fedeltà, la coerenza geometrica e la qualità visiva dei risultati ottenuti, premettendo il confronto tra approcci differenti e favorendo la riproducibilità delle analisi.

In particolare, il lavoro di *Fakhri Torkhani et al. (2015)* [106] costituisce un contributo rilevante nel panorama della valutazione percettiva della qualità delle mesh dinamiche. Lo studio propone un framework per la valutazione soggettiva e oggettiva della qualità percettiva di mesh 3D animate, evidenziando l'importanza di metriche capaci di riflettere con maggiore accuratezza la sensibilità umana alle distorsioni introdotte da operazioni lossy. Le metriche individuate nel paper sono state selezionate sulla base della loro capacità di discriminare tra diversi livelli e tipologie di distorsione (spaziale, temporale, compressione, errori di rete), e risultano quindi particolarmente adatte a supportare valutazioni robuste e comparabili delle prestazioni dei modelli.

In questo progetto, ci si concentra su un sottoinsieme di queste metriche, selezionato per la loro rilevanza e capacità predittiva. In particolare, si considerano la SPD, la SWSPD e la DMPD, che verranno descritte nel dettaglio nelle sezioni seguenti.

5.2.1 Misura di distorsione percettiva basata sulle caratteristiche spaziali

La prima metrica oggettiva, su cui si basano quelle che verranno analizzate in seguito, è quella che misura la distorsione percettiva della mesh basandosi sulle caratteristiche spaziali (*Spatial-based Perceptual Distortion measure*, **SPD**).

Tale indicatore si fonda sul concetto di *ruvidità locale*, una proprietà geometrica che riflette la complessità della superficie attorno a ciascun vertice, e definita come il valore assoluto del *Laplaciano della curvatura gaussiana*.

La distanza percettiva locale tra un vertice v_{ij}^r della mesh di riferimento M_r e il corrispondente vertice v_{ij}^d della mesh distorta M_d è calcolata come:

$$e_{ij}^s = \frac{|LR_{ij}^r - LR_{ij}^d|}{LR_{ij}^r + LR_{ij}^d + C_{LR}} \quad (5.1)$$

dove LR_{ij}^r e LR_{ij}^d rappresentano i valori di *ruvidità locale* nei due vertici, e $C_{LR} = 0.002$ è una costante introdotta per prevenire instabilità numeriche. Tale formulazione permette di modellare parzialmente l'effetto di *mascheramento visivo*: una stessa variazione di ruvidità è percepita in modo meno evidente in regioni già geometricamente complesse.

Le distanze locali e_{ij}^s vengono poi addizionate spazialmente attraverso una somma di Minkowski (con esponente $m_s = 3$) per ottenere la distanza percettiva per frame:

$$f_i^s = \left(\frac{1}{n_v} \sum_{j=1}^{n_v} (e_{ij}^s)^{m_s} \right)^{1/m_s} \quad (5.2)$$

Successivamente, le distanze per frame f_i^s vengono combinate temporalmente, sempre tramite una somma di Minkowski, per ottenere la **distanza percettiva globale puramente spaziale (SPD)**:

$$SPD = \left(\frac{1}{n_f} \sum_{i=1}^{n_f} (f_i^s)^{m_t} \right)^{1/m_t} \quad (5.3)$$

dove $m_t = 4$, n_v è il numero di vertici per frame, e n_f il numero totale di frame. La scelta degli esponenti m_s e m_t è motivata empiricamente: valori più alti danno maggior peso agli errori localmente o temporalmente più evidenti. Infatti, è stato ampiamente verificato attraverso che la qualità percepita dagli esseri umani è fortemente influenzata dalle parti maggiormente compromesse del contenuto visivo, piuttosto che da quelle integre. Pertanto, i valori scelti empiricamente dagli autori sono $m_s = 3$ e $m_t = 4$, poiché offrono risultati convincenti quando confrontati con i punteggi soggettivi su un ampio database.

I risultati sperimentali riportati nel paper mostrano che, nonostante la semplicità della definizione, la metrica SPD offre una buona correlazione con i punteggi soggettivi, in particolare per distorsioni puramente spaziali.

5.2.2 Misura di distorsione percettiva spaziale ponderata dalla velocità

La misura di distorsione percettiva spaziale ponderata dalla velocità (*Speed-Weighted Spatial-based Perceptual Distortion*, **SWSPD**) estende la metrica appena descritta (basata solo su distorsione spaziale) introducendo un importante fattore percettivo: la velocità dei vertici nel tempo. Questo approccio si basa sull'osservazione, supportata da prove sperimentali, che le distorsioni spaziali risultano meno percepibili nelle regioni ad alta velocità di movimento, mentre diventano più evidenti in aree statiche o quasi statiche.

Per modellare tale comportamento, la metrica introduce una ponderazione basata sulla velocità SP_{ij} di ciascun vertice v_{ij} . La velocità è definita come il vettore di spostamento del vertice tra due frame consecutivi, ed è normalizzata rispetto alla diagonale del *bounding box* del mesh, al fine di garantire l'invarianza rispetto alla

scala della scena.

La distorsione locale percettiva es_{ij} è quindi così definita, secondo un modello lineare a tratti:

$$e_{ij}^{st} = \begin{cases} w_1 \cdot es_{ij}, & \text{se } SP_{ij} \leq T_1 \\ \left[w_1 + \frac{w_2 - w_1}{T_2 - T_1} (SP_{ij} - T_1) \right] \cdot es_{ij}, & \text{se } T_1 < SP_{ij} \leq T_2 \\ w_2 \cdot es_{ij}, & \text{se } SP_{ij} > T_2 \end{cases} \quad (5.4)$$

dove i parametri empirici sono settati dagli autori del paper come $T_1 = 0.01$, $T_2 = 0.05$, $w_1 = 1.0$, $w_2 = 0.3$. Questi valori sono stati ottenuti mediante analisi sperimentali e ottimizzati per massimizzare la correlazione con le valutazioni soggettive, come riportato nei test effettuati su database costruiti ad hoc.

Successivamente, le distorsioni locali ponderate vengono addizionate spazialmente all'interno di ciascun frame attraverso una somma di Minkowski di ordine $m = 3$:

$$f_s^t(i) = \left(\sum_j (e_{ij}^{st})^m \right)^{1/m} \quad (5.5)$$

L'aggregazione temporale delle distorsioni per ciascun frame consiste in una somma di Minkowski di ordine $n = 4$:

$$SWSPD = \left(\frac{1}{n_f} \sum_{i=1}^{n_f} (f_s^t(i))^n \right)^{1/n} \quad (5.6)$$

In conclusione, SWSPD si dimostra una metrica più robusta in presenza di artefatti localizzati e dinamici come errori di trasmissione di rete o compressioni aggressive, mantenendo una maggiore coerenza con la percezione umana della qualità.

5.2.3 Integrazione delle caratteristiche temporali e della metrica DMPD

L'integrazione di nuove caratteristiche temporali all'interno della metrica precedentemente definita rappresenta un ulteriore passo avanti nella valutazione percettiva della qualità delle mesh dinamiche e costituisce un miglioramento ulteriore delle prestazioni. In particolare, vengono considerate due semplici caratteristiche temporali, rispettivamente legate alla velocità del vertice e alla direzione del movimento del vertice. Entrambe sono progettate per catturare variazioni dinamiche che sfuggono a misure puramente spaziali.

Dato un vertice v_{ij}^r nella mesh di riferimento M^r e il corrispondente vertice v_{ij}^d nella mesh distorta M^d , si definiscono i vettori di movimento (in avanti) $\vec{d}_{ij}^r$ e $\vec{d}_{ij}^d$, normalizzati rispetto alla diagonale del *bounding box* della scena, per garantire l'invarianza alla scala. Poiché non è possibile calcolare i vettori di movimento in avanti per l'ultimo frame, in quel caso si utilizza il vettore di movimento all'indietro.

Il contrasto basato sulla velocità, denominato *speed-based local perceptual distance*, si esprime come:

$$ss_{ij} = \frac{\left\| \vec{d}_{ij}^r \right\| - \left\| \vec{d}_{ij}^d \right\|}{\left\| \vec{d}_{ij}^r \right\| + \left\| \vec{d}_{ij}^d \right\| + C_S} \quad (5.7)$$

dove $C_S = 0.002$ è una costante per evitare instabilità numeriche.

In modo complementare, viene definito termine che valuta la variazione nella direzione del movimento. Si calcola come:

$$as_{ij} = \frac{AG_{ij}^r - AG_{ij}^d}{AG_{ij}^r + AG_{ij}^d + C_{AG}} \quad (5.8)$$

dove AG_{ij}^r e AG_{ij}^d sono gli angoli tra i vettori di movimento avanti e indietro per i vertici v_{ij}^r e v_{ij}^d , e $C_{AG} = \frac{1}{32\pi}$. Per i vertici nel primo e nell'ultimo frame, l'angolo viene copiato rispettivamente dal secondo e penultimo frame.

Le distanze locali ss_{ij} e as_{ij} vengono addizionate spazialmente e temporalmente tramite somme di Minkowski, utilizzando potenze rispettivamente pari a 5 nello spazio e 3 nel tempo, ottenendo così le distanze globali:

- **SPPD**: *Speed-based Perceptual Distance*
- **APD**: *Angle-based Perceptual Distance*

Infine, la metrica completa **DMPD** (*Dynamic Mesh Perceptual Distance*) viene definita come combinazione pesata delle tre distanze percettive:

$$DMPD = \sqrt{w_1 \cdot SWSPD^2 + w_2 \cdot SPPD^2 + w_3 \cdot APD^2} \quad (5.9)$$

dove $SWSPD$ rappresenta la metrica pesata sulla rugosità spaziale (precedentemente descritta), e i pesi sono fissati empiricamente a $w_1 = 0.4$, $w_2 = 0.5$ e $w_3 = 0.1$.

Tale formulazione offre prestazioni superiori nella correlazione con i punteggi soggettivi, confermando l'efficacia dell'integrazione di caratteristiche temporali nella valutazione della qualità percettiva delle mesh dinamiche. Tuttavia, gli autori sottolineano che la metrica è ancora migliorabile, soprattutto per quanto riguarda strategie di pooling spaziale e temporale più sofisticate e adattive.

5.2.4 Analisi delle metriche

Per valutare l'efficacia delle metriche descritte, all'interno di questo progetto di tesi sono state condotte analisi su un set di **10 mesh tridimensionali**, suddivise equamente tra soggetti maschili (5) e femminili (5). Ciascuna mesh è stata animata con una sequenza di camminata composta da 32 frame, utilizzando il servizio online *Mixamo* [113], che permette di generare facilmente animazioni realistiche a partire da modelli 3D statici.

Per ogni mesh sono stati considerati quattro diversi casi:

1. il solo corpo nudo fornito dalla rete LVD;
2. il corpo naked di LVD a cui è stata sostituita manualmente la testa con quella ricostruita attraverso DECA;
3. la mesh precedente con l'aggiunta di capelli modellati dalla rete HairStep;
4. la mesh al punto 2 vestita con t-shirt e pantaloni lunghi.

Le metriche **SPD**, **SWSPD** e **DMPD** sono state calcolate senza fare riferimento ad una mesh ground-truth (*reference mesh*), ma considerando la coerenza temporale della mesh tra frame successivi della sequenza animata. Le misure, quindi, non valutano la fedeltà rispetto a un modello statico di riferimento, ma si concentrano sull'analisi della stabilità e della regolarità del movimento nel tempo, utilizzando come parametri la curvatura, la velocità dei vertici e la direzione del movimento.

SPD	MESH 1F	MESH 2F	MESH 3F	MESH 4F	MESH 5F	MESH 1M	MESH 2M	MESH 3M	MESH 4M	MESH 5M
LVD	3,1684	3,0480	3,1408	3,1435	3,2469	3,0738	3,1004	2,9036	2,7613	2,8923
LVD + Deca	3,0164	3,0483	3,2435	2,9806	2,9848	2,9816	2,8964	3,1262	2,7851	2,9277
LVD + Deca + HairStep	3,0164	3,0483	2,9806	2,6627	2,9848	2,9816	2,8964	3,1262	2,8111	2,9277
LVD + Deca + clothes	2,6275	2,6189	2,6298	2,6236	2,5121	2,7201	2,5594	2,5526	2,5879	2,6557

Figura 5.1: Risultati del calcolo della metrica SPD.

Dai risultati riportati nella tabella sopra, si può notare come l'aggiunta degli abiti (caso 4) produce una riduzione dei valori della **metrica SPD** rispetto agli altri tre casi considerati. Questa tendenza è osservabile su tutte le mesh testate, indipendentemente dal sesso del soggetto, definendo quindi un miglioramento della coerenza temporale del movimento. In particolare, la mesh LVD vestita (t-shirt e pantaloni lunghi) e con testa DECA mostra i valori SPD più bassi tra tutte le configurazioni sperimentate, suggerendo che l'integrazione degli abiti abbia effetto stabilizzante sul comportamento dinamico dell'intera mesh. Tale comportamento può essere interpretato come una sorta di effetto di "smorzamento", che riduce le oscillazioni locali e i micro-movimenti incoerenti dei vertici del modello tra frame successivi, rendendo

il movimento percepito più fluido, regolare e realistico.

SWPD	MESH 1F	MESH 2F	MESH 3F	MESH 4F	MESH 5F	MESH 1M	MESH 2M	MESH 3M	MESH 4M	MESH 5M
LVD	3,1048	2,9846	3,0903	3,0799	3,2034	3,0162	3,0479	2,8708	2,7128	2,8383
LVD + Deca	3,0158	3,0589	3,3316	2,9813	3,0222	2,9655	2,9011	3,1496	2,7807	2,9159
LVD + Deca + HairStep	3,0158	3,0589	2,9813	2,6872	3,0222	2,9655	2,9011	3,1496	2,7873	2,9159
LVD + Deca + clothes	2,6108	2,6076	2,6233	2,6483	2,5291	2,7074	2,5516	2,5558	2,5649	2,6411

Figura 5.2: Risultati del calcolo della metrica SWPD.

Anche per la **metrica SWPD** si osserva un trend dei risultati: il quarto caso analizzato risulta nuovamente il più efficace nel ridurre i valori della metrica su tutte le mesh. Questa riduzione appare particolarmente evidente se confrontata con le configurazioni prive di abbigliamento, confermando che l'aggiunta degli abiti non rappresenta solo un miglioramento estetico, ma contribuisce anche a stabilizzare il comportamento dinamico della mesh in modo significativo.

La metrica SWPD, focalizzata sulle variazioni spaziali pesate tra frame successivi, è sensibile a micro-discontinuità e deformazioni locali incoerenti. In questo contesto, l'effetto "smorzante" indotto dall'abbigliamento risulta particolarmente vantaggioso, poiché agisce da vincolo aggiuntivo che limita deformazioni spurie e fluttuazioni irregolari, soprattutto nelle regioni di interfaccia e nelle articolazioni. I risultati quantitativi mostrano infatti un miglioramento uniforme sia per le mesh maschili che femminili, con un abbassamento complessivo dei valori SWPD che riflette un comportamento più armonico e regolare nel tempo.

DMPD	MESH 1F	MESH 2F	MESH 3F	MESH 4F	MESH 5F	MESH 1M	MESH 2M	MESH 3M	MESH 4M	MESH 5M
LVD	5,3876	5,1409	5,3673	5,3828	5,5387	5,2391	5,3042	4,9387	4,7427	4,9258
LVD + Deca	5,4701	5,4764	5,8888	5,3937	5,4271	5,3587	5,2970	5,6435	5,0973	5,2452
LVD + Deca + HairStep	5,4086	5,4219	5,3298	4,8639	5,3778	5,2841	5,2303	5,5799	5,0171	5,1742
LVD + Deca + clothes	5,4409	5,3054	5,3297	5,4328	5,2280	5,4308	5,2380	5,0976	5,1979	5,2903

Figura 5.3: Risultati del calcolo della metrica DMPD.

L'analisi della **metrica DMPD**, che valuta le deviazioni dinamiche tra frame successivi ed è particolarmente sensibile a disallineamenti nella traiettoria del movimento, evidenzia un comportamento leggermente più eterogeneo rispetto alle precedenti, confermando comunque il trend favorevole all'introduzione degli abiti (caso 4).

Pur non registrando sempre i valori minimi assoluti su tutte le mesh, la configurazione *LVD + Deca + clothes* ottiene in media risultati comparabili o migliori rispetto alle altre soluzioni, in particolare sulla maggior parte delle mesh femminili e maschili). L'abbigliamento continua dunque ad esercitare un effetto stabilizzante anche a livello dinamico, limitando brusche variazioni di posizione tra frame consecutivi.

Tuttavia, rispetto a SPD e SWPD, l'influenza degli abiti su DMPD risulta leggermente meno marcata e più soggetta a variazioni locali. Questo può essere attribuito al fatto che la metrica penalizza differenze dinamiche globali, dove l'intervento dell'abbigliamento può essere meno efficace in presenza di traiettorie articolari complesse o deformazioni non localizzate.

Al termine di questa analisi, nel confronto tra SPD, SWPD e DMPD, emerge una tendenza coerente: l'aggiunta degli abiti contribuisce a una maggiore regolarità spaziale e temporale del movimento. SPD e SWPD mostrano una riduzione chiara e uniforme dei valori, evidenziando come le mesh analizzate siano maggiormente stabili e coerenti a livello locale e strutturale. La DMPD conferma questo miglioramento anche sul piano dinamico, seppur con un effetto meno marcato.

L'inclusione degli abiti, quindi, non solo migliora l'aspetto visivo, ma ottimizza anche la qualità del movimento, rendendo la mesh più realistica e fluida, caratteristiche fondamentali per applicazioni nell'ambito dell'animazione, della simulazione o della AR/VR.

5.2.5 Metriche qualitative

Nel caso in cui si volesse procedere in futuro con un'analisi più approfondita dei risultati, è possibile considerare una serie di metriche qualitative e visive, utili per valutare la coerenza e il livello di realismo degli avatar generati:

- **Coerenza visiva complessiva:** valutazione generale dell'armonia del modello, analizzando la fusione tra corpo, testa e capelli, e l'assenza di discontinuità visibili tra le parti.
- **Realismo percettivo:** grado di naturalezza percepita nella forma e nei dettagli del modello, in particolare nelle proporzioni anatomiche, nelle espressioni facciali, nel comportamento dei capelli e nelle pieghe dei capi di abbigliamento.
- **Adattamento delle texture:** qualità dell'applicazione delle texture su ogni componente della mesh e coerenza cromatica tra le diverse regioni del modello.
- **Somiglianza con l'immagine originale:** livello di riconoscibilità del soggetto rappresentato rispetto all'immagine RGB utilizzata in input.
- **Compatibilità nell'utilizzo:** possibilità di utilizzare agevolmente l'output generato in ambienti 3D, per animazione o rendering, senza incorrere in artefatti, mesh aperte o errori strutturali e geometrici.

L'adozione di queste metriche qualitative consentirebbe di rafforzare la valutazione del sistema anche in assenza di una ground truth tridimensionale, fornendo un riscontro utile per il miglioramento futuro della pipeline.

5.3 Limiti del modello

Nonostante i risultati ottenuti siano promettenti, il sistema presenta alcuni limiti che potrebbero influenzare l'accuratezza della ricostruzione e la variabilità degli avatar generati.

Il primo aspetto critico riguarda sicuramente il fatto che al momento non si è trovato un algoritmo adatto ed efficace per la sostituzione della testa generata dalla rete LVD con quella più accurata ottenuta tramite DECA. Sono stati analizzati diversi approcci per effettuare tale sostituzione direttamente tramite script in Blender, ma i risultati presentavano frequentemente artefatti geometrici e discontinuità visibili, come buchi nella mesh, difficili da correggere in modo automatico. Il miglioramento della qualità richiederebbe un intervento manuale dell'utente, operazione incompatibile con la natura automatizzata della pipeline proposta in questo progetto di tesi. Per questo motivo, come già spiegato nel Capitolo 4, si è deciso di mantenere le teste originali fornite dalla rete LVD, così da evitare che eventuali imperfezioni compromettessero l'integrità delle mesh e la coerenza nel calcolo delle metriche di valutazione.

Il fatto che il modello riceva in input una singola immagine RGB, senza informazioni di profondità o viste multiple costituisce una limitazione del sistema, che si riflette soprattutto sulla ricostruzione del corpo e dei capelli, poiché la rete ha accesso esclusivamente alla parte frontale del soggetto, rendendo difficile l'interpretazione delle geometrie posteriori. Anche per una rete neurale per la ricostruzione automatica dei vestiti tale aspetto potrebbe costituire un problema: tuttavia, l'attuale approccio basato su garments standardizzati, modellati a mano e non, non lo rende un limite. Modificare il sistema, in modo che accetti in input anche più immagini multi-view o brevi sequenze video, potrebbe migliorare notevolmente la qualità della ricostruzione, permettendo una visione a 360° del soggetto. Ciò porterebbe però a sostituire le attuali reti utilizzate, progettate per lavorare su una singola immagine.

Il sesso del soggetto è un altro aspetto problematico del progetto: come già spiegato nel capitolo 4 di questa trattazione, la rete LVD che si occupa della ricostruzione del corpo modella le mesh del body di genere neutro. Per far sì che le shilouettes in output abbiano forme caratterizzanti maschili o femminili sarebbe opportuno effettuare nuovamente la fase di training della rete.

Un ulteriore vincolo riguarda la varietà degli indumenti (vestiti e scarpe) applicabili. Attualmente, il modello consente di vestire l'avatar solo con le categorie di abiti predefinite, limitando la possibilità di generare outfit maggiormente diversificati. Per superare questo limite, sarebbe necessario ampliare il numero di modelli di garments, includendo più capi realizzati manualmente per la cloth simulation, o

implementando una strategia per automatizzare l'estrazione degli abiti da dataset dedicati. Vi sarebbe quindi necessità di script più robusti per il draping, in modo da gestire con precisione maggiore la vestibilità dei garments sulla mesh del corpo. Problema analogo per le texture applicate ai vestiti e alle scarpe: attualmente, infatti, il sistema si limita ad assegnare colori RGB ai materiali in Blender, senza una vera e propria estrazione delle texture dall'immagine di input. Per ottenere un risultato più fedele, si potrebbe integrare una rete pre-addestrata per la generazione automatica delle texture o progettare un metodo più accurato per estrarre direttamente i pattern e i dettagli tessili dalla fotografia di input, preservandone il realismo.

Un altro limite riguarda la fase di preprocessing e segmentazione del soggetto. Attualmente, il sistema utilizza un tool online per la rimozione dello sfondo, che risulta preciso anche in condizioni di illuminazione critiche e background disturbanti. Come ulteriore implementazione del progetto, sarebbe opportuno utilizzare soluzioni "interne" senza dover ricorrere a strumenti esterni per il preprocessing delle immagini utilizzate. Si potrebbe inoltre implementare la ricostruzione del modello anche in caso di occlusioni sul soggetto, andando a prevederne la mesh.

Infine, un'ultima area di miglioramento riguarda l'attuale metodo utilizzato per vestire l'avatar. Al momento, il sistema si affida totalmente all'intervento dell'utente che va a selezionare manualmente e a suo piacimento i capi di abbigliamento da posizionare sopra il modello umanoide. A tal proposito, una soluzione più efficiente potrebbe essere lo sviluppo di un primo modulo image-to-text, integrato direttamente nel progetto, in grado di analizzare l'immagine in input e restituire una descrizione dettagliata dei vestiti presenti. Tale descrizione verrà poi processata dall'add-on, che andrà a determinare automaticamente quali capi di abbigliamento utilizzare per restituire un risultato il più fedele possibile all'input originale. Questo permetterebbe, in un primo momento, di eliminare l'incertezza dettata dall'azione dell'utente sul sistema con il rischio di lasciare l'avatar completamente o parzialmente nudo.

Questi limiti rappresentano delle opportunità di miglioramento che, se affrontate, potrebbero rendere il sistema più preciso, versatile e adattabile a un maggior numero di casi d'uso.

Capitolo 6

Conclusioni e sviluppi futuri

6.1 Conclusioni

Il progetto presentato in questa tesi ha dimostrato la possibilità di generare avatar 3D a partire da una singola immagine RGB in-the-wild, andando a concentrarsi maggiormente sulla modellazione accurata della forma del corpo del soggetto piuttosto che sulla sua posa. A differenza di altre soluzioni orientate all'animazione o alla personalizzazione dell'avatar, questo approccio permette una ricostruzione fedele della morfologia del soggetto, a cui viene imposta una posizione in T-pose per garantire una base uniforme per l'elaborazione delle mesh e per ovviare ad errori di ricostruzione.

Per la modellazione del corpo, la pipeline utilizza la rete LVD, mentre per la ricostruzione dei capelli viene adottato dal progetto HairStep. La struttura della testa, invece, viene generata tramite DECA, permettendo di ottenere un modello 3D che presenta una maggiore precisione nei dettagli facciali. La gestione del vestiario avviene tramite una prima interazione dell'utente con il sistema e una seconda fase di scripting algoritmico in Blender, un approccio che non si basa su reti neurali, quindi, ma su regole definite per la modellazione e il drappeggio degli abiti, consentendo un elevato controllo sulla forma e l'aderenza del vestiario alla mesh corporea.

Durante la fase di sperimentazione, è emerso come la qualità della ricostruzione dipenda fortemente dalle caratteristiche dell'immagine di input (in particolare, per il posizionamento delle braccia e della testa del soggetto). Sono stati individuati alcuni requisiti fondamentali che l'immagine deve rispettare per garantire una generazione fedele e coerente della mesh. Se tali criteri non vengono soddisfatti, possono verificarsi errori strutturali o artefatti, compromettendo la qualità del modello finale.

Un ulteriore aspetto riguarda la gestione delle texture. Ad eccezione di quella del viso, elaborata direttamente dalla rete DECA, le altre vengono estratte manualmente e applicate alla mesh tramite la creazione di materiali in Blender a cui vengono

attribuiti valori RGB definiti. Questo processo, pur costituendo una soluzione per ottenere maggiore realismo nel modello in output, può introdurre imprecisioni nei dettagli, specialmente in assenza di un metodo automatico per la generazione di texture più complesse.

In conclusione, questa tesi ha sviluppato un metodo per la generazione di avatar 3D statici, per cui nella successiva sezione di questo capitolo verranno evidenziate alcuni sviluppi futuri per un miglioramento della pipeline. L'approccio sviluppato si inserisce in un contesto di ricerca in cui gli avatar stanno assumendo un ruolo sempre più centrale, con un crescente interesse verso la loro applicazione in diversi settori. La metodologia proposta potrebbe quindi essere impiegata nei molteplici ambiti in cui le mesh umanoidi trovano usualmente applicazione, dalla realtà virtuale alla moda digitale, fino alle interfacce interattive e alle simulazioni immersive.

6.2 Sviluppi futuri

Il rapido progresso nel campo delle reti neurali per la ricostruzione 3D offre la possibilità di migliorare continuamente l'approccio adottato in questa tesi, mantenendo la struttura di base della pipeline ma aggiornando costantemente le metodologie impiegate. L'integrazione di nuovi progetti più avanzati potrebbe portare a una maggiore accuratezza nella generazione dell'output, riducendone le limitazioni e migliorando la qualità complessiva della ricostruzione.

Prima di tutto, sarebbe utile proporre una soluzione più accurata per rendere gli avatar animabili. E' già stato compiuto un passo in questa direzione, applicando un'armatura riggata alla mesh del body, che consente la manipolazione della posa e l'animazione sia del corpo che dei vestiti. L'utilizzo del progetto verrebbe così esteso rendendolo adatto anche in caso di animazione 3D, gaming e realtà virtuale, oltre alla semplice generazione di avatar statici.

A tal proposito potrebbe essere aumentato il focus del sistema anche sulla ricostruzione della posa, attualmente trattata in modo secondario rispetto alla modellazione della forma del corpo, integrando nel sistema metodologie avanzate per catturare con precisione la posizione del soggetto ed eliminando il vincolo della posa standard in input.

Un altro possibile sviluppo riguarda la gestione del vestiario, che attualmente viene ricostruito in due fasi distinte (intervento dell'utente e scripting in Blender). Si propone quindi di adottare, in un primo momento, una rete image-to-text che automatizzi in un unico step il processo di vestizione dell'avatar con i capi base presenti attualmente nel progetto ed eliminando così l'intervento dell'utente. Successivamente, si potrebbe pensare di integrarla con una seconda rete, più sofisticata e dedicata

completamente alla generazione di outfit, permettendo di evitare limitazioni nella modellazione dei garments e delle loro texture, offrendo una maggiore coerenza visiva e ampliando le possibilità di personalizzazione da parte dell'utente. A tal proposito, l'incremento del numero di garments e di textures disponibili darebbero la possibilità all'utente finale di scegliere tra svariate opzioni di outfit e materiali in base alle preferenze personali. Questo renderebbe il sistema più flessibile e adatto a un'ampia gamma di applicazioni.

Per tutti gli aspetti appena descritti, potrebbe essere necessario integrare i modelli preesistenti con dataset specifici per ogni rete utilizzata (dataset di pose, di abbigliamento, di espressioni facciali, di acconciature ecc.). L'aumento del numero di reference ed esempi a disposizione dei progetti consentirebbe di incrementare la varietà delle ricostruzioni possibili e di migliorare la capacità del modello di generalizzare su soggetti con caratteristiche diverse.

Infine, sarebbe utile definire delle metriche visive per valutare il realismo e la coerenza dell'output con l'immagine in input.

Questi sviluppi potrebbero rendere il sistema ancora più accurato, flessibile e applicabile in ambiti come la moda digitale, l'intrattenimento, la realtà virtuale e la creazione di contenuti interattivi, contribuendo all'evoluzione degli avatar 3D come strumenti sempre più avanzati e versatili.

Bibliografia

- [1] Blender, <https://www.blender.org/>
- [2] Python, <https://www.python.org/>
- [3] AI Describe Tool, <https://imageprompt.org/describe-image>
- [4] Pytorch, <https://pytorch.org/>
- [5] Pycharm, <https://www.jetbrains.com/pycharm/>
- [6] Anaconda, <https://docs.anaconda.com/anaconda/>
- [7] BlenderKit, <https://www.blenderkit.com/>
- [8] AI Background Remover, <https://aibackgroundremover.org/>
- [9] Bingyang Zhou and Haoyu Zhou and Tianhai Liang and Qiaojun Yu and Siheng Zhao and Yuwei Zeng and Jun Lv and Siyuan Luo and Qiancai Wang and Xinyuan Yu and Haonan Chen and Cewu Lu and Lin Shao, *ClothesNet: An Information-Rich 3D Garment Model Repository with Simulated Clothes Environment*, IEEE/CVF International Conference on Computer Vision (ICCV), 2023. <https://sites.google.com/view/clothesnet>
- [10] Bharat Lal Bhatnagar, Cristian Sminchisescu, Christian Theobalt and Gerard Pons-Moll, *Combining Implicit Function Learning and Parametric Models for 3D Human Reconstruction*, Max Planck Institute for Informatics, Saarland Informatics Campus, Germany, ECCV 2020. <http://virtualhumans.mpi-inf.mpg.de/ipnet/>
- [11] Loper, Matthew and Mahmood, Naureen and Romero, Javier and Pons-Moll, Gerard and Black, Michael J., *SMPL: A Skinned Multi-Person Linear Model*, ACM Trans. Graphics (Proc. SIGGRAPH Asia), Oct 2015. <https://smpl.is.tue.mpg>

de./

- [12] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas and M. J. Black, *Expressive Body Capture: 3D Hands, Face, and Body from a Single Image*, Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2019, Long Beach, CA. <https://smpl-x.is.tue.mpg.de/>
- [13] Osman, Ahmed A A and Bolkart, Timo and Black, Michael J., *STAR: A Sparse Trained Articulated Human Body Regressor*, European Conference on Computer Vision (ECCV), 2020. <https://star.is.tue.mpg.de>
- [14] Sungho Chun, Sungbum Park, Ju Yong Chan, *Learnable Human Mesh Triangulation for 3D Human Pose and Shape Estimation*, Dept of ECE, Kwangwoon University, Korea (NCSOFT, Korea), 2023. <https://github.com/MPI-IS/mesh>
- [15] Choutas, Vasileios and Müller, Lea and Huang, Chun-Hao P. and Tang, Siyu and Tzionas, Dimitrios and Black, Michael J., *Accurate 3D Body Shape Regression using Metric and Semantic Attributes*, Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2022. <https://shapy.is.tue.mpg.de/>
- [16] Zhang, Hongwen and Tian, Yating and Zhang, Yuxiang and Li, Mengcheng and An, Liang and Sun, Zhenan and Liu, Yebin, *PyMAF-X: Towards Well-aligned Full-body Model Regression from Monocular Images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023. <https://www.liuyebin.com/pymaf-x/>
- [17] Zhang, Hongwen and Tian, Yating and Zhou, Xinchu and Ouyang, Wanli and Liu, Yebin and Wang, Limin and Sun, Zhenan, *PyMAF: 3D Human Pose and Shape Regression with Pyramidal Mesh Alignment Feedback Loop*, Proceedings of the IEEE International Conference on Computer Vision, 2021. <https://www.liuyebin.com/pymaf-x/>
- [18] Li, Jiefeng and Bian, Siyuan and Liu, Qi and Tang, Jiasheng and Wang, Fan and Lu, Cewu, *NIKI: Neural Inverse Kinematics with Invertible Neural Networks for 3D Human Pose and Shape Estimation*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2023. <https://arxiv.org/abs/2305.08590>
- [19] Tripathi, Shashank and Müller, Lea and Huang, Chun-Hao P. and Taheri Omid and Black, Michael J. and Tzionas, Dimitrios, *3D Human Pose Estimation via Intuitive Physics*, Conference on Computer Vision and Pattern Recognition (CVPR), 2023. <https://ipman.is.tue.mpg.de>

- [20] Corona, Enric and Pons-Moll, Gerard and Alenyà, Guillem and Moreno-Noguer, Francesc, *Learned Vertex Descent: A New Direction for 3D Human Model Fitting*, ECCV, 2022. <https://www.iri.upc.edu/people/ecorona/lvd/>
- [21] Pang, Hui En and Cai, Zhongang and Yang, Lei and Qingyi, Tao and Zhonghua, Wu and Zhang, Tianwei and Liu, Ziwei, *Towards Robust and Expressive Whole-body Human Pose and Shape Estimation*, S-Lab, Nanyang Technological University, NeurIPS, 2023. <https://arxiv.org/abs/2312.08730>
- [22] Keller, Marilyn and Werling, Keenon and Shin, Soyong and Delp, Scott and Pujades, Sergi and Liu, C. Karen and Black, Michael J., *From Skin to Skeleton: Towards Biomechanically Accurate 3D Digital Humans*, ACM Transaction on Graphics (ToG), December 2023. <https://doi.org/10.1145/3618381>
- [23] Lu Dai, Liqian Ma, Shenhan Qian, Hao Liu, Ziwei Liu, Hui Xiong, *Cloth2Body: Generating 3D Human Body Mesh from 2D Clothing*, The Hong Kong University of Science and Technology (Guangzhou), 2023. https://openaccess.thecvf.com/content/ICCV2023/papers/Dai_Cloth2Body_Generating_3D_Human_Body_Mesh_from_2D_Clothing_ICCV_2023_paper.pdf
- [24] Rohan Sarkar, Achal Dave, Gerard Medioni, and Benjamin Biggs, *Shape of You: Precise 3D shape estimations for diverse body types*, Purdue University, Amazon, 2023. https://openaccess.thecvf.com/content/CVPR2023W/CVFAD/papers/Sarkar_Shape_of_You_Precise_3D_Shape_Estimations_for_Diverse_Body_CVPRW_2023_paper.pdf
- [25] Nikolaos Zioulis, James F. O’Brien, *KBody: Towards general, robust, and aligned monocular whole-body estimation*, Klothed Technologies Inc., UC Berkeley, 2023. https://openaccess.thecvf.com/content/CVPR2023W/RHOBIN/papers/Zioulis_KBody_Towards_General_Robust_and_Aligned_Monocular_Whole-Body_Estimation_CVPRW_2023_paper.pdf
- [26] Zhao, Qitao and Zheng, Ce and Liu, Mengyuan and Wang, Pichao and Chen, Chen, *PoseFormerV2: Exploring Frequency Domain for Efficient and Robust 3D Human Pose Estimation*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2023. <https://github.com/QitaoZhao/PoseFormerV2>
- [27] Abdelreheem, Ahmed and Skorokhodov, Ivan and Ovsjanikov, Maks and Wonka, Peter, *SATR: Zero-Shot Semantic Segmentation of 3D Shapes*, Proceedings of the International Conference on Computer Vision (ICCV), 2023. <https://samir55.github.io/SATR/>

- [28] Shi-Min Hu and Zheng-Ning Liu and Meng-Hao Guo and Junxiong Cai and Jiahui Huang and Tai-Jiang Mu and Ralph R. Martin, *Subdivision-based Mesh Convolution Networks*, 2022. <https://doi.org/10.1145/3506694>
- [29] Bhatnagar, Bharat Lal and Tiwari, Garvita and Theobalt, Christian and Pons-Moll, Gerard, *Multi-Garment Net: Learning to Dress 3D People from Images*, IEEE International Conference on Computer Vision (ICCV), October 2019. <https://virtualhumans.mpi-inf.mpg.de/mgn/>
- [30] Liao, Tingting and Zhang, Xiaomei and Xiu, Yuliang and Yi, Hongwei and Liu, Xudong and Qi, Guo-Jun and Zhang, Yong and Wang, Xuan and Zhu, Xiangyu and Lei, Zhen, *High-Fidelity Clothed Avatar Reconstruction from a Single Image*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2023. <https://arxiv.org/abs/2304.03903>
- [31] Huang, Yangyi and Yi, Hongwei and Liu, Weiyang and Wang, Haofan and Wu, Boxi and Wang, Wenxiao and Lin, Binbin and Zhang, Debing and Cai, Deng, *One-shot Implicit Animatable Avatars with Model-based Priors*, IEEE Conference on Computer Vision (ICCV), 2023. <https://huangyangyi.github.io/ELICIT/>
- [32] Feng, Yao and Yang, Jinlong and Pollefeys, Marc and Black, Michael J. and Bolkart, Timo, *Capturing and Animation of Body and Clothing from Monocular Video*, SIGGRAPH Asia 2022 Conference Papers, 2022. <https://yfeng95.github.io/scarf/>
- [33] Patel, Chaitanya and Liao, Zhouyingcheng and Pons-Moll, Gerard, *TailorNet: Predicting Clothing in 3D as a Function of Human Pose, Shape and Garment Style*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2020. <https://virtualhumans.mpi-inf.mpg.de/tailornet/>
- [34] Zhao, Fuwei and Xie, Zhenyu and Kampffmeyer, Michael and Dong, Haoye and Han, Songfang and Zheng, Tianxiang and Zhang, Tao and Liang, Xiaodan, *M3D-VTON: A Monocular-to-3D Virtual Try-On Network*, Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2021. <https://github.com/fyviezhao/M3D-VTON>
- [35] Aymen Mir, Thiemo Alldieck, Gerard Pons-Moll, *Learning to Transfer Texture from Clothing Images to 3D Humans*, Max Planck Institute for Informatics, Saarland Informatics Campus, Germany, Computer Graphics Lab, TU Braunschweig, Germany, arXiv:2003.02050v2, March 2020. <https://arxiv.org/pdf/2003.02050>

- [36] De Luigi, Luca and Li, Ren and Guillard, Benoit and Salzmman, Mathieu and Fua, Pascal, *DrapeNet: Garment Generation and Self-Supervised Draping*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023. <https://liren2515.github.io/page/drapenet/drapenet.html>
- [37] Ren, Li and Guillard, Benoit and Remelli, Edoardo and Fua, Pascal, *DIG: Draping Implicit Garment over the Human Body*, Asian Conference on Computer Vision, 2022. <https://liren2515.github.io/page/dig/dig.html>
- [38] Zhang, Hongwen and Lin, Siyou and Shao, Ruizhi and Zhang, Yuxiang and Zheng, Zerong and Huang, Han and Guo, Yandong and Liu, Yebi, *CloSET: Modeling Clothed Humans on Continuous Surface with Explicit Template Decomposition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2023. <https://www.liuyebin.com/closet/>
- [39] Grigorev, Artur and Thomaszewski, Bernhard and Black, Michael J and Hilgig, Otmar, *HOOD: Hierarchical Graphs for Generalized Modelling of Clothing Dynamics*, Computer Vision and Pattern Recognition (CVPR), 2023. <https://dolorousrtur.github.io/hood/>
- [40] Bharat Lal Bhatnagar Garvita Tiwari Christian Theobalt Gerard Pons-Moll, *Multi-Garment Net: Learning to Dress 3D People from Images*, Max Planck Institute for Informatics, Saarland Informatics Campus, Germany. <https://virtualhumans.mpi-inf.mpg.de/papers/bhatnagar2019mgn/bhatnagar2019mgn.pdf>
- [41] Igor Santesteban, Miguel A. Otaduy, and Dan Casas, *SNUG: Self-Supervised Neural Dynamic Garments*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. <http://mslab.es/projects/SNUG/>
- [42] Bertiche, Hugo and Madadi, Meysam and Escalera, Sergio, *Neural Cloth Simulation*, Association for Computing Machinery, December 2022.
url : <https://doi.org/10.1145/3550454.3555491>, doi :10.1145/3550454.3555491.
<https://hbertiche.github.io/NeuralClothSim/>
- [43] Bertiche, Hugo and Madadi, Meysam and Escalera, Sergio, *PBNS: Physically Based Neural Simulation for Unsupervised Garment Pose Space Deformation*, Association for Computing Machinery, December 2021.
url : <https://doi.org/10.1145/3478513.3480479>,
doi : 10.1145/3478513.3480479.
<https://hbertiche.github.io/PBNS/>
- [44] Moon, Gyeongsik and Nam, Hyeongjin and Shiratori, Takaaki and Lee, Ky-

oung Mu, *3D Clothed Human Reconstruction in the Wild*, European Conference on Computer Vision (ECCV), 2022. https://github.com/hygenie1228/ClothWild_RELEASE

[45] Bertiche, Hugo and Madadi, Meysam and Tylson, Emilio and Escalera, Sergio, *DeePSD: Automatic deep skinning and pose space deformation for 3D garment animation*, Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021. <https://hbertiche.github.io/DeePSD/>

[46] A. Revkov and D. Kanin, *FITTIN - Online 3D Shoe Try-on*, Proc. of 3DBODY.TECH 2020 - 11th Int. Conf. and Exh. on 3D Body Scanning and Processing Technologies, Online/Virtual, Nov. 2020, <https://doi.org/10.15221/20.58>. <https://proc.3dbody.tech/abstracts/2020/2058revkov.html>

[47] Antić, Dimitrije and Tiwari, Garvita and Ozcomlekci, Batuhan and Marin, Riccardo and Pons-Moll, Gerard, *CloSe: A 3D Clothing Segmentation Dataset and Model*, International Conference on 3D Vision (3DV), March 2024. <https://virtualhumans.mpi-inf.mpg.de/close3dv24/>

[48] Luo, Zhongjin and Liu, Haolin and Li, Chenghong and Du, Wanghao and Jin, Zirong and Sun, Wanhu and Nie, Yinyu and Chen, Weikai and Han, Xiaoguang, *GarVerseLOD: High-Fidelity 3D Garment Reconstruction from a Single In-the-Wild Image using a Dataset with Levels of Details*, ACM Transactions on Graphics (TOG), 2024. https://garverselod.github.io/?utm_source=chatgpt.com

[49] Liu, Xin and Li, Jiancheng and Wang, Jiaqi and Liu, Ziwei, *MMFashion: An Open-Source Toolbox for Visual Fashion Analysis*, ACM Multimedia 2021, Open Source Software Competition, 2021. <https://arxiv.org/abs/2005.08847>

[50] Liu, Lijuan and Xu, Xiangyu and Lin, Zhijie and Liang, Jiabin and Yan, Shui-cheng, *Towards Garment Sewing Pattern Reconstruction from a Single Image*, ACM Transactions on Graphics (SIGGRAPH Asia), 2023. <https://arxiv.org/abs/2311.04218v1>

[51] Radu Alexandru Rosu and Shunsuke Saito and Ziyang Wang and Chenglei Wu and Sven Behnke and Giljoo Nam, *Neural Strands: Learning Hair Geometry and Appearance from Multi-View Images*, ECCV 2022. https://radualexandru.github.io/neural_strands/

[52] Sklyarova, Vanessa and Chelishev, Jenya and Dogaru, Andreea and Medvedev, Igor and Lempitsky, Victor and Zakharov, Egor, *Neural Haircut: Prior-Guided Strand-Based Hair Reconstruction*, IEEE International Conference on Computer Vi-

sion (ICCV), 2023. <https://samsunglabs.github.io/NeuralHaircut/>

[53] Currius, R. R. and Assarsson, U. and Sintorn, E., *Real-Time Hair Filtering with Convolutional Neural Networks*, Proceedings of the ACM on Computer Graphics and Interactive Techniques, Proc. ACM Comput. Graph. Interact. Tech., May 2022.

doi: 10.1145/3522606. <https://www.cse.chalmers.se/~ramon/hair-filtering.html>

[54] Wang, Ziyang and Nam, Giljoo and Stuyck, Tuur and Lombardi, Stephen and Cao, Chen and Saragih, Jason and Zoll, *NeuWigs: A Neural Dynamic Model for Volumetric Hair Capture and Animation*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2023.

<https://ziyanw1.github.io/neuwigs/>

[55] Yi Zhou, Liwen Wu, Jun Xing, Weikai Chen, Han-Wei Kung, Xin Tong, Hao Li, *HairNet: Single-View Hair Reconstruction using Convolutional Neural Networks*, ECCV 2018. <https://vgl.ict.usc.edu/Research/HairNet/>

[56] Menglei Chai, Tianjia Shao, Hongzhi Wu, Yanlin Weng, Kun Zhou, *Auto-Hair: Fully Automatic Hair Modeling from A Single Image*, Zhejiang University. <http://kunzhou.net/zjugaps/autohair/>

[57] Zheng, Yujian and Jin, Zirong and Li, Moran and Huang, Haibin and Ma, Chongyang and Cui, Shuguang and Han, Xiaoguang, *Hairstep: Transfer synthetic to real using strand and depth maps for single-view 3d hair modeling*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023. <https://paulyzheng.github.io/research/hairstep/>

[58] Wu, Keyu and Ye, Yifan and Yang, Lingchen and Fu, Hongbo and Zhou, Kun and Zheng, Youyi, *NeuralHDHair: Automatic High-fidelity Hair Modeling from a Single Image Using Implicit Neural Representations*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.

https://openaccess.thecvf.com/content/CVPR2022/papers/Wu_NeuralHDHair_Automatic_High-Fidelity_Hair_Modeling_From_a_Single_Image_Using_CVPR_2022_paper.pdf

Yang, Lingchen and Shi, Zefeng and Zheng, Youyi and Zhou, Ku, *Dynamic hair modeling from monocular videos using deep neural networks*, ACM Transactions on Graphics (TOG), 2019.

[59] Zhang, Meng and Zheng, Youyi, *Hair-GAN: Recovering 3D hair structure from a single image using generative adversarial networks*, Visual Informatics, 2019, El-

sevier. <https://arxiv.org/abs/1811.06229>

[60] Hugo Bertiche, Meysam Madadi and Sergio Escaleral, *CLOTH3D: Clothed 3D Humans*, European Conference on Computer Vision, 2020.

<https://arxiv.org/pdf/1912.02792v2>

[61] Tiwari, Garvita and Bhatnagar, Bharat Lal and Tung, Tony and Pons-Moll, Gerard, *SIZER: A Dataset and Model for Parsing 3D Clothing and Learning Size Sensitive 3D Clothing*, European Conference on Computer Vision (ECCV), August 2020. <https://virtualhumans.mpi-inf.mpg.de/sizer/>

[62] Heming, Zhu and Yu, Cao and Hang, Jin and Weikai, Chen and Dong, Du and Zhangye, Wang and Shuguang, Cui and Xiaoguang, Han, *Deep Fashion3D: A Dataset and Benchmark for 3D Garment Reconstruction from Single Images*, Computer Vision ECCV 2020, Springer International Publishing. <https://github.com/GAP-LAB-CUHK-SZ/deepFashion3D>

[63] Bingyang Zhou and Haoyu Zhou and Tianhai Liang and Qiaojun Yu and Siheng Zhao and Yuwei Zeng and Jun Lv and Siyuan Luo and Qiancai Wang and Xinyuan Yu and Haonan Chen and Cewu Lu and Lin Shao, *ClothesNet: An Information-Rich 3D Garment Model Repository with Simulated Clothes Environment*, IEEE/CVF International Conference on Computer Vision (ICCV), 2023.

<https://sites.google.com/view/clothesnet>

[64] He, Kai and Yao, Kaixin and Zhang, Qixuan and Yu, Jingyi and Liu, Lingjie and Xu, Lan, *DressCode: Autoregressively Sewing and Generating Garments from Text Guidance*, ACM Transactions on Graphics (TOG), 2024. URL: <https://ihe-kaii.github.io/DressCode/>

[65] Yuying Ge and Ruimao Zhang and Lingyun Wu and Xiaogang Wang and Xiaoou Tang and Ping Luo, *A Versatile Benchmark for Detection, Pose Estimation, Segmentation and Re-Identification of Clothing Images*, CVPR 2019.

<https://github.com/switchablenorms/DeepFashion2>

[66] Xingxing Zou, Xintong Han, Waikeng Wong, *CLOTH4D: A Dataset for Clothed Human Reconstruction*, Laboratory for Artificial Intelligence in Design, Huya Inc., School of Fashion and Textiles, The Hong Kong Polytechnic University, 2023.

https://openaccess.thecvf.com/content/CVPR2023/papers/Zou_CLOTH4D_A_Dataset_for_Clothed_Human_Reconstruction_CVPR_2023_paper.pdf

[67] Pietro Musoni and Simone Melzi and Umberto Castellani, *GIM3D plus: A labeled 3D dataset to design data-driven solutions for dressed humans*,

doi:<https://doi.org/10.1016/j.gmod.2023.101187>,
url:<https://www.sciencedirect.com/science/article/pii/S1524070323000176>.
<https://github.com/PietroMsn/GIM3D>

[68] Yang, Haotian and Zhu, Hao and Wang, Yanru and Huang, Mingkai and Shen, Qiu and Yang, Ruigang and Cao, Xun, *FaceScape: A Large-Scale High Quality 3D Face Dataset and Detailed Riggable 3D Face Prediction*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020. <https://github.com/zuhao-nju/facescape>

[69] Martyniuk, Tetiana and Kupyn, Orest and Kurlyak, Yana and Krasheniyi, Igor and Matas, Jiří and Sharmanska, Viktoriia, *DAD-3DHeads: A Large-scale Dense, Accurate and Diverse Dataset for 3D Head Alignment from a Single Image*, Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2022. <https://github.com/PinataFarms/DAD-3DHeads>

[70] Yang, Zhitao and Cai, Zhongang and Mei, Haiyi and Liu, Shuai and Chen, Zhaoxi and Xiao, Weiye and Wei, Yukun and Qing, Zhongfei and Wei, Chen and Dai, Bo and Wu, Wayne and Qian, Chen and Lin, Dahua and Liu, Ziwei and Yang, Lei, *SynBody: Synthetic Dataset with Layered Human Models for 3D Human Perception and Modeling*, Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2023. <https://synbody.github.io/>

[71] Black, Michael J. and Patel, Priyanka and Tesch, Joachim and Yang, Jinlong, *BEDLAM: A Synthetic Dataset of Bodies Exhibiting Detailed Lifelike Animated Motion*, Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), June 2023. <https://bedlam.is.tue.mpg.de/>

[72] Bai, Haoran and Kang, Di and Zhang, Haoxian and Pan, Jinshan and Bao, Linchao, *FFHQ-UV: Normalized Facial UV-Texture Dataset for 3D Face Reconstruction*, IEEE Conference on Computer Vision and Pattern Recognition, June 2023. <https://github.com/csbhr/FFHQ-UV>

[73] Patel, Priyanka and Huang, Chun-Hao P. and Tesch, Joachim and Hoffmann, David T. and Tripathi, Shashank and Black, Michael J., *AGORA: Avatars in Geography Optimized for Regression Analysis*, Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), June 2021. <https://agora.is.tue.mpg.de/>

[74] Huang, Chun-Hao P. and Yi, Hongwei and Höschle, Markus and Safroshkin, Matvey and Alexiadis, Tsvetelina and Polikovsky, Senya and Scharstein, Daniel and Black, Michael J., *Capturing and Inferring Dense Full-Body Human-Scene Contac*,

Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), June 2022. <https://rich.is.tue.mpg.de/index.html>

[75] Taewoo Kim, Chaeyeon Chung, Sunghyun Park, Gyojung Gu, Keonmin Nam, Wonzo Choe, Jaesung Lee, Jaegul Choo, *K-Hairstyle: A Large-scale Korean hairstyle dataset for virtual hair editing and hairstyle classification*, arXiv:2102.06288v3, October 2021. <https://psh01087.github.io/K-Hairstyle/>

[76] Yutao Chen, Yuxuan Zhang, Zhongrui Huang, Zhenyao Luo, and Jinpeng Chen, *CelebHair: A New Large-Scale Dataset for Hairstyle Recommendation based on CelebA*, arXiv:2104.06885v1, April 2021. <https://arxiv.org/pdf/2104.06885>

[77] Yu Deng and Jiaolong Yang and Sicheng Xu and Dong Chen and Yunde Jia and Xin Tong, *Accurate 3D Face Reconstruction with Weakly-Supervised Learning: From Single Image to Image Set*, IEEE Computer Vision and Pattern Recognition Workshops, 2019. https://github.com/sicxu/Deep3DFaceRecon_pytorch

[78] Dib, Abdallah and Hafemann, Luiz Gustavo and Got, Emeline and Anderson, Trevor and Fadaeinejad, Amin and Cruz, Rafael M. O. and Carbonneau, Marc-Andr, *MoSAR: Monocular Semi-Supervised Model for Avatar Reconstruction using Differentiable Shading*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2024. <https://ubisoft-laforge.github.io/character/mosar/>

[79] Chai, Zenghao and Zhang, Tianke and He, Tianyu and Tan, Xu and Baltrusaitis, Tadas and Wu, HsiangTao and Li, Runnan and Zhao, Sheng and Yuan, Chun and Bian, Jiang, *HiFace: High-Fidelity 3D Face Reconstruction by Learning Static and Dynamic Details*, Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2023. <https://project-hiface.github.io/>

[80] Kirschstein, Tobias and Giebenhain, Simon and Niessner, Matthias, *DiffusionAvatars: Deferred Diffusion for High-fidelity 3D Head Avatars*, arXiv preprint arXiv:2311.18635, 2023. <https://tobias-kirschstein.github.io/diffusion-avatars/>

[81] Wojciech Zielonka and Timo Bolkart and Justus Thies, *Towards Metrical Reconstruction of Human Faces*, European Conference on Computer Vision, 2022. <https://zielon.github.io/mica/>

[82] Biwen Lei and Jianqiang Ren and Mengyang Feng and Miaomiao Cui and Xuansong Xie, *A Hierarchical Representation Network for Accurate and Detailed Face Reconstruction from In-The-Wild Images*, 2023. <https://younglbw.github.io>

[io/HRN-homepage/](https://github.com/HRN-homepage/)

- [83] Luo Jiang, Juyong Zhang, Bailin Deng, Hao Li, and Ligang Liu, *3D Face Reconstruction with Geometry Details from a Single Image*, arXiv:1702.05619v2, June 2018. <https://arxiv.org/pdf/1702.05619>
- [84] Anh Tuan Tran, Tal Hassner, Iacopo Masi, and Gerard Medioni, *Regressing Robust and Discriminative 3D Morphable Models with a very Deep Neural Network*, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Honolulu, 2017. https://talhassner.github.io/home/publication/2017_CVPR
- [85] Wood, Erroll and Baltrusaitis, Tadas and Hewitt, Charlie and Dziadzio, Sebastian and Cashman, Thomas J and Shotton, Jamie, *Fake it till you make it: face analysis in the wild using synthetic data alone*, Proceedings of the IEEE/CVF international conference on computer vision, 2021.
URL: <https://microsoft.github.io/FaceSynthetics/>
- [86] Raman, Chirag and Hewitt, Charlie and Wood, Erroll and Baltrusaitis, Tadas, *Mesh-Tension Driven Expression-Based Wrinkles for Synthetic Faces*, Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2023. URL: <https://microsoft.github.io/DynamicWrinkles/>
- [87] Aaron S. Jackson, Adrian Bulat, Vasileios Argyriou, Georgios Tzimiropoulos, *Large Pose 3D Face Reconstruction from a Single Image via Direct Volumetric CNN Regression*, The University of Nottingham, UK. https://vision.unipv.it/CV/materiale2017-18/fifthchoice/Jackson_Large_Pose_3D_ICCV_2017_paper.pdf
- [88] Feng, Yao and Feng, Haiwen and Black, Michael J. and Bolkart, Timo, *Learning an Animatable Detailed 3D Face Model from In-the-Wild Images*, ACM Transactions on Graphics (ToG), Proc. SIGGRAPH, August 2021. <https://deca.is.tue.mpg.de/>
- [89] Chen, Anpei and Chen, Zhang and Zhang, Guli and Mitchell, Kenny and Yu, Jingyi, *Photo-Realistic Facial Details Synthesis from Single Image*, Proceedings of the IEEE International Conference on Computer Vision, 2019. https://apchenstu.github.io/facial_details/
- [90] Yajing Chen, Fanzi Wu, Zeyu Wang, Yibing Song, Yonggen Ling, and Linchao Bao, *Self-supervised Learning of Detailed 3D Face Reconstruction*, arXiv:1910.11791v2, September 2020. <https://arxiv.org/pdf/1910.11791>
- [91] Ling, Jingwang and Wang, Zhibo and Lu, Ming and Wang, Quan and Qian,

Chen and Xu, Feng, *Structure-aware Editable Morphable Model for 3D Facial Detail Animation and Manipulation*, Proceedings of the European Conference on Computer Vision (ECCV), 2022. <https://arxiv.org/abs/2207.09019>

[92] Xuan Gao and Chenglai Zhong and Jun Xiang and Yang Hong and Yudong Guo and Juyong Zhang, *Reconstructing Personalized Semantic Facial NeRF Models From Monocular Video*, ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia), 2022. <https://ustc3dv.github.io/NeRFBlendShape/>

[93] Bharadwaj, Shrisha and Zheng, Yufeng and Hilliges, Otmar and Black, Michael J. and Abrevaya, Victoria Fernandez, *FLARE: Fast learning of Animatable and Relightable Mesh Avatars*, ACM Transactions on Graphics, December 2023, doi: <https://doi.org/10.1145/3618401>. <https://flare.is.tue.mpg.de/>

[94] Li, Xueting and De Mello, Shalini and Liu, Sifei and Nagano, Koki and Iqbal, Umar and Kautz, Jan, *Generalizable One-shot Neural Head Avatar*, Arxiv 2023. <https://research.nvidia.com/labs/lpr/one-shot-avatar/>

[95] Li, Junxuan and Saito, Shunsuke and Simon, Tomas and Lombardi, Stephen and Li, Hongdong and Saragih, Jason, *MEGANE: Morphable Eyeglass and Avatar Network*, arXiv preprint arXiv:2302.04868, 2023. <https://junxuan-li.github.io/megane/>

[96] Zhongyuan Zhao, Zhenyu Bao, Qing Li, Guoping Qiu, Kanglin Liu, *PSAvatar: A Point-based Shape Model for Real-Time Head Avatar Animation with 3D Gaussian Splatting*, arXiv:2401.12900v5, June 2024. <https://arxiv.org/pdf/2401.12900>

[97] Xu, Yuelang and Chen, Benwang and Li, Zhe and Zhang, Hongwen and Wang, Lizhen and Zheng, Zerong and Liu, Yebin, *Gaussian Head Avatar: Ultra High-fidelity Head Avatar via Dynamic Gaussians*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. <https://yuelangx.github.io/gaussianheadavatar/>

[98] Bao, Chong and Zhang, Yinda and Li, Yuan and Zhang, Xiyu and Yang, Bang-bang and Bao, Hujun and Pollefeys, Marc and Zhang, Guofeng and Cui, Zhaopeng, *GeneAvatar: Generic Expression-Aware Volumetric Head Avatar Editing from a Single Image*, The IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR), 2024. <https://zju3dv.github.io/geneavatar/>

[99] Zeng, Xiaoxing and Peng, Xiaojiang and Qiao, Yu, *DF2Net: A Dense-Fine-Finer Network for Detailed 3D Face Reconstruction*, Proceedings of the IEEE International Conference on Computer Vision, 2019. <https://openaccess.thecvf>.

[com/content_ICCV_2019/papers/Zeng_DF2Net_A_Dense-Fine-Finer_Network_for_Detailed_3D_Face_Reconstruction_ICCV_2019_paper.pdf](https://openaccess.thecvf.com/content_ICCV_2019/papers/Zeng_DF2Net_A_Dense-Fine-Finer_Network_for_Detailed_3D_Face_Reconstruction_ICCV_2019_paper.pdf)

[100] Gecer, Baris and Ploumpis, Stylianos and Kotsia, Irene and Zafeiriou, Stefanos, *GANFIT: Generative Adversarial Network Fitting for High Fidelity 3D Face Reconstruction*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019. <https://github.com/barisgecer/ganfit>

[101] Cong Wang, Di Kang, He-Yi Sun, Shen-Han Qian, Zi-Xuan Wang, Linchao Bao, Song-Hai Zhang, *MeGA: Hybrid Mesh-Gaussian Head Avatar for High-Fidelity Rendering and Head Editing*, arXiv:2404.19026v1, April 2024. <https://arxiv.org/pdf/2404.19026>

[102] Ramon, Eduard and Triginer, Gil and Escur, Janna and Pumarola, Albert and Garcia, Jaime and Giro-i-Nieto, Xavier and Moreno-Noguer, Francesc, *H3D-Net: Few-Shot High-Fidelity 3D Head Reconstruction*, H3D-Net: Few-Shot High-Fidelity 3D Head Reconstruction, 2021. <https://crisalixsa.github.io/h3d-net/>

[103] Yao Feng and Fan Wu and Xiaohu Shao and Yanfeng Wang and Xi Zhou, *Joint 3D Face Reconstruction and Dense Alignment with Position Map Regression Network*, ECCV 2018. <https://github.com/yfeng95/PRNet>

[104] Saito, Shunsuke and Yang, Jinlong and Ma, Qianli and Black, Michael J., *SCANimate: Weakly Supervised Learning of Skinned Clothed Avatar Networks*, Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), June 2021. <https://scanimate.is.tue.mpg.de/>

[105] Jae Shin Yoon, Kihwan Kim, Jan Kautz, Hyun Soo Park, *Neural 3D Clothes Retargeting from a Single Image*, arXiv:2102.00062v1, January 2021. <https://arxiv.org/pdf/2102.00062>

[106] Fakhri Torkhani, Kai Wang, Jean-Marc Chassery, *Perceptual Quality Assessment of 3D Dynamic Meshes: Subjective and Objective Studies*, 2015. <https://www.sciencedirect.com/science/article/abs/pii/S0923596514001866>

[107] Lei, Jiahui and Sridhar, Srinath and Guerrero, Paul and Sung, Minhyuk and Mitra, Niloy and Guibas, Leonidas J., *Pix2Surf: Learning Parametric 3D Surface Models of Objects from Images*, Proceedings of European Conference on Computer Vision (ECCV), August 2020. <https://geometry.stanford.edu/projects/pix2surf>

[108] Mingjin Chen and Junhao Chen and Xiaojun Ye and Huan-ang Gao and Xiao-

xue Chen and Zhaoxin Fan and Hao Zhao, *Ultraman: Single Image 3D Human Reconstruction with Ultra Speed and Detail*, 2024. <https://air-discover.github.io/Ultraman/>

[109] Ubuntu, <https://www.ubuntu-it.org//>

[110] Pumarola, Albert and Sanchez, Jordi and Choi, Gary and Sanfeliu, Alberto and Moreno-Noguer, Francesc, *3DPeople: Modeling the Geometry of Dressed Humans*, International Conference in Computer Vision (ICCV), 2019. <https://www.albertpumarola.com/research/3DPeople/index.html>

[111] Zhang, Hongwen and Tian, Yating and Zhou, Xinchu and Ouyang, Wanli and Liu, Yebin and Wang, Limin and Sun, Zhenan, *PyMAF: 3D Human Pose and Shape Regression with Pyramidal Mesh Alignment Feedback Loop*, Proceedings of the IEEE International Conference on Computer Vision, 2021. <https://www.liuyebin.com/pymaf-x/>

[112] Debnath, Pronay, et al., *Celeb-FBI: A Benchmark Dataset on Human Full Body Images and Age, Gender, Height and Weight Estimation using Deep Learning Approach*, 2024. <https://arxiv.org/pdf/2407.03486/>

[113] Mixamo, <https://www.mixamo.com/>

Elenco delle figure

1.1	Esempi di avatar bidimensionali.	6
1.2	Esempi di avatar tridimensionali.	6
1.3	Esempio di applicazione di Virtual try-on.	8
1.4	Esempi di avatar full body per applicazioni computer-based.	9
1.5	Suddivisione dei metodi di ricostruzione 3D.	9
1.6	Tassonomia dei metodi di ricostruzione 3D.	10
1.7	Ricostruzione 3D con PIFu e Tex-PIFu a partire da un'immagine RGB.	11
1.8	Schema del rendering volumetrico basato su input 5D, con trasformazione in colore e densità e ottimizzazione tramite rendering loss.	12
1.9	Confronto tra NeRF e Gaussian Splatting per la sintesi 3D.	13
1.10	Pipeline di Gaussian Splatting: da point cloud a rendering.	14
1.11	Architettura di una GAN.	15
1.12	Rappresentazione della pipeline di CLIP.	17
1.13	Confronto tra le mesh crate con CLIP e con DreamField.	18
2.1	Differenza tra i metodi model-based e model-free.	21
2.2	Esempio di ricostruzione con il metodo SMPL.	23
2.3	Esempi di implementazioni SMPL.	24
2.4	Stima della posa da parte del metodo Poseformer	25
2.5	Rappresentazione della pipeline di ricostruzione per i metodi PBS.	26
2.6	Rappresentazione della pipeline di ricostruzione per i metodi data-driven.	27
2.7	Pipeline di lavoro del metodo DRAPENET.	28
2.8	Pipeline di trasferimento della texture in Pix2Surf.	30
2.9	Metodo MGN per la ricostruzione delle animazioni.	32
2.10	Pipeline per la ricostruzione del volto di HiFace.	35
2.11	Esempio di wrinke map detection.	35
2.12	Esempio del metodo GeneAvatar.	37
2.13	Segmentazione della chioma effettuata dal metodo AutoHair.	40
2.14	Esempio di modellazione basata su ciocche della rete Neural Hair Strands.	41

2.15	Esempi di corpi in posa presenti nel dataset Bedlam.	42
2.16	Esempio tratto dal dataset DeepFashion3D.	43
2.17	Pipeline di adattamento dei vestiti adottata da SIZER.	44
2.18	Dati presenti nel database FaceScape.	45
2.19	Tintura virtuale per immagini del dataset K-Hairstyle	46
2.20	Immagini estratte dal dataset CelebHair.	46
3.1	Confronto tra i metodi di ricostruzione del corpo.	48
3.2	Pipeline della rete LVD.	50
3.3	Esempi di ricostruzione dei vestiti con Clothwild.	51
3.4	Esempi di ricostruzione dei vestiti con Clothwild.	51
3.5	Interfaccia del tool AI Describe Image.	52
3.6	Pipeline di ricostruzione di un volto in DECA.	54
3.7	Pipeline di ricostruzione dei capelli di HairStep.	56
3.8	Esempio di segmentazione di una mesh umana con SATR.	57
3.9	Esempio di immagini presenti nel dataset Celeb-FBI.	58
3.10	Interfaccia di Anaconda Navigator per l'installazione di pacchetti. . .	62
3.11	Interfaccia di Pycharm.	63
3.12	Interfaccia del tool AI Background Remover.	64
4.1	Pipeline tematica del progetto.	65
4.2	Pipeline dettagliata del progetto.	66
4.3	Esempio di immagini utilizzate come input.	67
4.4	Immagini croppate per la rete DECA (a sinistra) e HairStep (a destra). .	68
4.5	Immagini cui è stato rimosso il background con il tool AI Back- ground Remover.	69
4.6	Immagini in input alla rete LVD.	70
4.7	Visualizzazione dei giunti del body SMPL.	72
4.8	Ricostruzione LVD di un corpo femminile.	73
4.9	Ricostruzione LVD di un corpo maschile.	74
4.10	Visualizzazione dei landmarks del viso.	76
4.11	Mesh di teste create con la rete DECA.	77
4.12	Textures del viso derivanti dalla rete DECA.	77
4.13	Chiome in formato .ply ottenute da HairStep.	79
4.14	Mesh toolkit add-on.	80
4.15	Scheletro adattato alla T-pose.	82
4.16	Corpi LVD a cui è stato applicato lo scheletro.	83
4.17	Sostituzione della testa DECA sul corpo LVD.	84
4.18	Posizionamento dei capelli sul corpo LVD.	85
4.19	Modelli di vestiti per la parte superiore del body.	87
4.20	Modelli di vestiti per la parte inferiore del body.	87
4.21	Modelli di scarpe presenti nel progetto.	91
4.22	Alcuni esempi di avatar completi.	93

5.1	Risultati del calcolo della metrica SPD.	99
5.2	Risultati del calcolo della metrica SWPD.	100
5.3	Risultati del calcolo della metrica DMPD.	100