Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Matematica

Superfici: Rappresentazione, Visualizzazione, Stampa e Navigazione Virtuale

Tesi di Laurea in Matematica

Relatore:
Chiar.ma Prof.ssa
Serena Morigi

Presentata da:
Achille Mariani

Anno Accademico 2023/2024

Cadendo, la goccia scava la pietra,
non per la sua forza, ma per la sua costanza.

Introduzione

La computer graphics si basa su concetti matematici e algoritmi per creare e manipolare immagini digitali. Le principali fasi coinvolte sono: la modellazione geometrica - creazione di oggetti tridimensionali (modelli 3D) utilizzando software specifici, il rendering - processo che trasforma un modello 3D in un'immagine 2D, simulando gli effetti della luce e dell'ombra, l'animazione - creazione di sequenze di immagini che danno vita a un oggetto o a una scena, l'interazione - consente all'utente di interagire con gli oggetti grafici, come nei videogiochi o nelle simulazioni. La computer graphics ha rivoluzionato numerosi settori: da settori artistici, come il gaming e la produzione cinematografica, a settori tecnici come l'ingegneria, l'architettura, la simulazione scientifica e la medicina. L'avanzamento di questo settore è strettamente legato allo sviluppo delle strutture hardware e delle capacità di calcolo dei computer, che hanno permesso la realizzazione di contenuti visivi sempre più realistici e complessi. La modellazione geometrica è una parte fondamentale della computer graphics. Si occupa della creazione di rappresentazioni matematiche di oggetti tridimensionali. In altre parole, è il processo di dare forma a un oggetto virtuale definendo le sue caratteristiche geometriche, come punti, linee, curve e superfici.

Questa tesi si propone di fornire una panoramica sui metodi e sulle tecniche con cui vengono trattate le superfici nella modellazione geometrica. La tesi è strutturata in una parte più teorica, affiancata dalla descrizione dell'attività sperimentale che è stata svolta presso il laboratorio VisLab del Dipartimento di Matematica. Nella prima parte si introducono i concetti di rappresentazione matematica e visualizzazione di superfici, ovvero si presentano le forme matematiche più utilizzate in computer graphics e le tecniche di visualizzazione delle superfici. Nella seconda parte si illustrano esempi di superfici generate utilizzando i metodi introdotti, mettendo quindi in pratica le conoscenze teoriche acquisite nella prima parte. In questa seconda parte, oltre alla rappresentazione al computer di superfici, vengono trattate anche la navigazione virtuale e la stampa 3D. Queste ultime due sono applicazioni della computer graphics ben note e ormai diffuse che supportano la produzione permettendo la prototipazione nella progettazione industriale

e la navigazione di ambienti virtuali. Per quanto riguarda la navigazione di una scena virtuale viene considerato un esempio semplice di un ambiente museale che espone le superfici create, e permette di esplorare alcune funzionalità dei visori 3D. La stampa 3D viene esaminata più nel dettaglio e, stampando alcune delle superfici presentate nei primi capitoli, vengono trattate le criticità e alcune tecniche per risolvere questi problemi della stampa 3D.

Le sperimentazioni condotte nel corso di questa tesi hanno dimostrato l'efficacia degli strumenti matematici e algoritmici studiati, evidenziando come tali tecniche siano in grado di gestire in maniera efficiente la rappresentazione e la manipolazione delle superfici. Questi risultati, oltre a confermare la validità dei metodi utilizzati, offrono spunti interessanti per ulteriori approfondimenti, in particolare per quanto riguarda l'ottimizzazione degli algoritmi nel rendering di superfici ad alta risoluzione e la loro applicazione in ambiti specifici come la simulazione fisica e la stampa 3D. Questi sviluppi potrebbero aprire la strada a nuove possibilità applicative sia in ambito industriale che accademico.

In conclusione, questa tesi si propone di esplorare i fondamenti teorici e pratici alla base della modellazione geometrica in computer graphics, con particolare attenzione al trattamento delle superfici. Nonostante la trattazione semplice che utilizza concetti elementari della computer graphics, la tesi ha portato risultati sperimentali interessanti e promettenti.

Indice

Capitolo 1

Rappresentazione di superfici

1.1 Introduzione

Le varietà/manifold sono fondamentali per comprendere la struttura dello spazio e per descrivere fenomeni fisici. Consentono di studiare oggetti con curvature e topologie complesse, che non possono essere descritti semplicemente utilizzando lo spazio euclideo.
Un **k-manifold** (o **k-varietà**) è uno spazio topologico M tale che per ogni punto $p \in M$, esiste un intorno aperto $U \subset M$ e un omeomorfismo $\varphi : U \to V \subset \mathbb{R}^k$, dove V è un aperto in $\mathbb{R}^k$.
Una superficie è un 2-manifold immerso in $\mathbb{R}^3$ con o senza boundary.
Un esempio di superficie senza boundary è la sfera di raggio unitario $S^2 = \{(x, y, z) \in \mathbb{R}^3 | x^2 + y^2 + z^2 = 1\}$, mentre un esempio di superficie con boundary è il disco $D^2 = \{(x, y, 0) \in \mathbb{R}^3 | x^2 + y^2 \leq 1\}$.
Lo studio della rappresentazione di superfici riguarda la ricerca di forme matematiche con cui descrivere le superfici. Analizzeremo due tipi di rappresentazione: *implicita* ed *esplicita*. Tra le rappresentazioni esplicite, considereremo quelle parametriche, di particolare interesse nella modellazione geometrica e nella computer graphics.
Le superfici **esplicite parametriche** vengono definite tramite la valutazione di una funzione a valori vettoriali $f : \Omega \subset \mathbb{R}^2 \to S = f(\Omega) \subset \mathbb{R}^3$. La rappresentazione **implicita**, invece, individua le superfici come luogo degli zeri di funzioni del tipo $f : \mathbb{R}^3 \to \mathbb{R}$, ottenendo dunque $S = \{x \in \mathbb{R}^3 | f(x) = 0\}$.

Esempio 1.1. Possiamo rappresentare matematicamente una sfera di raggio unitario sia in forma parametrica tramite la funzione a valori vettoriali

$$f : [0, 2\pi] \times [0, \pi] \to \mathbb{R}^3, f(u, v) = \begin{pmatrix} \cos(u)\sin(v) \\ \sin(u)\sin(v) \\ \cos(v) \end{pmatrix},$$

sia in forma implicita tramite la funzione

$$F : \mathbb{R}^3 \to \mathbb{R}, F(x, y, z) = x^2 + y^2 + z^2 - 1.$$

□

Osservazione 1.2. La scelta del tipo di rappresentazione tiene in conto la facilità di rappresentazione, l'eventuale tipo di modellazione richiesta in seguito (ad esempio locale o globale) e il tipo di visualizzazione che si vorrà utilizzare.

1.2 Rappresentazione Parametrica

Come menzionato in precedenza, una superficie parametrica è l'immagine di una funzione 2-dimensionale in $\mathbb{R}^3$.

Definizione 1.3 (Superficie parametrica)**.** *Sia $\Omega \subset \mathbb{R}^2$ dominio parametrico. $S \subset \mathbb{R}^3$ è una superficie parametrica definita da una feunzione f a valori vettoriali $f : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3$ con*

$$S : f(u, v) = \begin{pmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{pmatrix}.$$

dove $x(u, v)$, $y(u, v)$, $z(u, v)$ sono funzioni $C^1(\Omega)$.

Qualora la topologia della superficie renda impossibile la rappresentazione tramite una sola funzione definita su un solo dominio, si ricorre all'utilizzo di più funzioni definite su più domini parametrici differenti.
Le superfici classiche, ossia quelle note come la sfera, il toro o altre superfici più complesse possono essere espresse in forma parametrica. Per rappresentare superfici a forma libera si possono utilizzare patch di Bézier o superfici spline.

Esempio 1.4. Il toro in forma esplicita parametrica è rappresentato dalla funzione

$$f : [0, 2\pi] \times [0, 2\pi] \to \mathbb{R}^3, f(u, v) = \begin{pmatrix} (R + r\cos v)\cos u \\ (R + r\cos v)\sin u \\ r\sin v \end{pmatrix}.$$

1.2.1 Superfici Spline e NURBS

Superfici a forma libera possono essere rappresentate mediante rappresentazione spline prodotto tensoriale, che si ottiene dalla teoria sulle curve spline, introdotte in Appendice

B . In questa trattazione si considerano le spline come approssimanti di forma, in generale il loro utilizzo nel CAD e nel calcolo scientifico è molto piu' ampio.

Definizione 1.5 (**Superficie spline**). *Siano U e V due intervalli chiusi e limitati, $\Delta_x = \{x_i\}_{i=1}^{k}$ e $\Delta_y = \{y_i\}_{i=1}^{h}$ le rispettive partizioni. Dati i punti di controllo $P_{ij} \in \mathbb{R}^3$, la superficie spline prodotto tensioriale in forma parametrica relativa alla partizione nodale $\Delta_x \times \Delta_y$ del dominio $U \times V$ è definita da*

$$S(u,v) = \sum_{i=1}^{n+K+1} \sum_{j=1}^{m+H+1} P_{ij} N_{i,n}(u) N_{j,m}(v),$$

dove $N_{i,n}(u), N_{j,m}(v)$ sono le B-spline di grado n e m, con partizioni estese Δ_x^* e K e H sono definiti come nella definizione B.3.

Analogamente possono essere definite le superfici NURBS tramite la formula

$$S(u,v) = \frac{\sum_{i=1}^{n+K+1} \sum_{j=1}^{m+H+1} w_{ij} P_{ij} N_{i,n}(u) N_{j,m}(v)}{\sum_{i=1}^{n+K+1} \sum_{j=1}^{m+H+1} w_{ij} N_{i,n}(u) N_{j,m}(v)}.$$

Vediamo dunque che le superfici spline sono un tipo particolare di superfici parametriche che godono però di tutte le proprietà di approssimazione delle superfici elencate per le curve spline. Questo è vero perché, se si fissa un parametro tra u e v, il prodotto tensoriale è esattamente una curva spline.
Esempi di superfici spline vengono riportati nella sezione 3.2, in cui vengono esplorate anche tecniche di modellazione come la costruzione di superfici cross-sectional.

Superfici di Bézier

Le curve di Bézier sono curve parametriche polinomiali costruite a partire da polinomi di Bernstein, cioè polinomi del tipo

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad t \in [0,1], \quad i = 0, 1, \ldots, n,$$

dove n è il grado del polinomio. Un esempio di curva di Bézier di grado 3 è

$$\mathbf{B}(t) = (1-t)^3 \mathbf{P}_0 + 3t(1-t)^2 \mathbf{P}_1 + 3t^2(1-t)\mathbf{P}_2 + t^3 \mathbf{P}_3,$$

con $t \in [0,1]$ e $\mathbf{P}_i \in \mathbb{R}^3$ i punti di controllo. In questo elaborato, però, introduciamo le curve di Bézier solo come caso particolare delle spline. Infatti, una curva spline coincide con una curva di Bézier di grado n a tratti se per la partizione estesa $\Delta^* = \{t_i\}_{i=1}^{2n+K+2}$ dell'intervallo $[a,b]$ valgono:

- $t_i = a$ per i primi $n+1$ nodi,

- $t_i = b$ per gli ultimi $n + 1$ nodi,
- $t_i = t_{i+k}$ per $k = 1, \ldots, n - 1$ e $i = 2n + 2 + jn$, cioè tutti i nodi interni sono ripetuti n volte.

Considerando poi il prodotto tensoriale di curve spline di questo tipo, cioè di curve di Bézier a tratti, si ottiene una superficie che può essere vista come patch di Bézier a tratti, dunque le superfici di Bézier sono un particolare sottoinsieme delle superfici spline.

1.3 Rappresentazione Implicita

Una superficie definita implicitamente consiste in un insieme di punti nello spazio che soddisfano una condizione rappresentata matematicamente dalla funzione $F : \mathbb{R}^3 \to \mathbb{R}$. Questa funzione associa ad ogni punto dello spazio un valore scalare e permette di raggruppare questi punti in base al valore che gli è stato assegnato.
Per definizione, se $F(x, y, z) = 0$, il punto $(x, y, z) \in \mathbb{R}^3$ è sulla superficie e di conseguenza F caratterizza intrinsecamente un volume: i punti per cui vale $F(x, y, z) < 0$ sono da una parte (per convenzione all'interno) della superficie, e quelli per cui vale $F(x, y, z) > 0$ sono dall'altra (per convenzione all'esterno).
I tipi di rappresentazione implicita vengono distinti per tipologia di funzione utilizzata. Tra le più note ed utilizzate ci sono le funzioni algebriche e le *Signed Distance Function* (SDF).

Rappresentazione algebrica (o polinomiale)

Con il termine *algebrica* si intende che l'equazione implicita F è un polinomio di grado arbitrario in x, y, z. Oltre alle classiche superfici note, ci sono forme molto interessanti che possono essere definite con polinomi di grado relativamente basso.

Esempio 1.6. Si può ottenere una superficie a forma di cuore, definita implicitamente dall'equazione $(2x^2 + y^2 + z^2 - 1)^3 + (0.1x^2 + y^2)z^3 = 0$, che è un polinomio di grado 5.

Esempio 1.7. La rappresentazione del toro dell'esempio 1.4 può essere fatta anche in forma implicita tramite l'equazione $\left(\sqrt{x^2 + y^2} - R\right)^2 + z^2 - r^2 = 0$. □

Signed Distance Function (SDF)

La *signed distance function* è un particolare tipo di funzione implicita in cui F rappresenta la distanza euclidea con segno dalla superficie, dove il segno indica la posizione nello spazio rispetto alla superficie come nell'interpretazione introdotta sopra.

Matematicamente, la SDF è definita mediante la funzione distanza $d : \mathbb{R}^3 \to \mathbb{R}$ da una superficie S con

$$d(x) = \pm dist(x, S) = \pm min_{y \in S}\{\|x - y\|\}.$$

Ovviamente $d(x) = 0$ quando il punto x è sulla superficie S.

Esempio 1.8. La funzione implicita $F(x) = d(x, 0) - 1$, con $x \in \mathbb{R}^3$ e d la funzione distanza, definisce implicitamente la sfera di raggio 1. La funzione $F(x) = d(x, S) - 1, x \in \mathbb{R}^3$ e S la circonferenza nel piano xy di raggio 5, definisce un toro di raggio minore 1 e raggio maggiore 5.

Le SDF sono particolarmente utili nella computer graphics e nella visualizzazione scientifica perché consentono una rappresentazione implicita di forme complesse. Per visualizzare una superficie definita da una SDF, si può campionare la funzione in un reticolo di punti e poi utilizzare algoritmi di rendering per visualizzare le iso-superfici.

Skeletal design

Un particolare tipo di rappresentazione che utilizza le SDF è quella che utilizza lo *skeletal design*. Lo skeleletal design prevede, data una superficie, di definire lo scheletro come mediana dei suoi punti[1] e poi di ri-definire la superficie come luogo dei punti a distanza fissata dalla scheletro. La mediana dei punti di una superficie può essere un punto, un segmento o una superficie. Il procedimento può essere fatto anche da zero, generando una superficie partendo proprio dalla definizione del suo scheletro.

Esempio 1.9. Lo scheletro di una sfera è il suo centro. Lo scheletro di un'ellissoide è un segmento lungo il suo semi-asse maggiore.

L'utilità principale della rappresentazione implicita tramite SDF tramite lo scheletro della superficie è quella di semplificare la struttura per permettere l'animazione della superficie stessa. Supponendo, ad esempio, di avere una superficie statica a forma di mano e volendola animare, è possibile definirla implicitamente tramite il suo scheletro, che è una struttura unidimensionale e quindi più facile da muovere nello spazio, e poi, determinando i movimenti del solo scheletro, i punti sulla superficie della mano si muoveranno di conseguenza. È stato testato [JB] che questa tecnica riesce a riprodurre fedelmente le pieghe che si generano nella pelle di una mano reale.

1.3.1 Reti neurali: Multilayer Perceptron e SIREN

Quando la rappresentazione implicita di una superficie non è disponibile, ma si dispone di dati (punti nello spazio e associate normali ai punti) sufficienti è possibile ricostruirla.

[1] Eventualmente sezionando la superficie e considerandone un sottoinsieme per volta.

Una tecnica recentemente introdotta che si è mostrata particolarmente efficace è quella che utilizza reti neurali MLP (introduzione alle reti neurali di tipo MLP in Appendice A) per approssimare la funzione SDF che rappresenta implicitamente la superficie.

La grande potenzialità di queste reti è ben espressa dal teorema di Cybenko di rappresentazione universale [C] che garantisce che una rete neurale possa approssimare una qualsiasi funzione continua.

Definizione 1.10. *Diciamo che σ è discriminante se, data una misura $\mu \in M(I_n)$*

$$\int_{I_n} \sigma(y^T x + \theta)\, d\mu(x) = 0$$

per ogni $y \in \mathbb{R}^n$ e $\theta \in \mathbb{R}$, implica che $\mu = 0$.

Definizione 1.11. *Diciamo che σ è sigmoidale se*

$$\sigma(t) \to \begin{cases} 1 & \text{per } t \to +\infty, \\ 0 & \text{per } t \to -\infty. \end{cases}$$

Teorema 1.12 (**Approssimazione universale per le reti neurali**)**.** *Sia σ una funzione continua e discriminante. Allora somme finite della forma*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j)$$

sono dense in $C(I_n)$, con $I_n = [0,1]^n$. In altre parole, dato $f \in C(I_n)$ e $\varepsilon > 0$, esiste una somma $G(x)$, della forma sopra indicata, per la quale

$$|G(x) - f(x)| < \varepsilon \quad \text{per ogni } x \in I_n.$$

Osservazione 1.13. Nonostante il teorema garantisca una soluzione arbitrariamente vicina alla funzione continua da approssimare, non è specificato il numero di neuroni necessari, che spesso potrebbe essere troppo grande.

Osservazione 1.14. Il teorema è stato dimostrato da Cybenko per funzioni di attivazione sigmoidali, ma nell'articolo [C] viene specificato che il risultato si può estendere anche a funzioni sinusoidali.

SIREN

Nel 2020 è stato pubblicato un articolo [S] sullo studio di una rete neurale MLP chiamata SIREN (SInusoidal REpresentation Network) per la ricostruzione della Signed Distance

Function a partire da nuvole di punti. Le funzioni di attivazionein SIREN sono periodiche, ossia sinusouidali. Questa scelta di funzioni di attivazione permette di approssimare la funzione cercata con una derivabile infinite volte e di conseguenza, almeno in teoria, in grado di approssimare derivata prima e seconda della funzione originale in maniera precisa.
A differenza del MLP, i pesi iniziali devono essere scelti con grande cura e ad hoc per ogni problema: nell'articolo [S] sono inseriti i parametri di ogni problema che hanno trattato. SIREN è in realtà una rete neurale in grado di risolvere anche altri problemi oltre alla ricostruzione di SDF di superfici 3D: può essere applicato a ricostruzione di segnali complessi (immagini, video, segnali audio), ma anche alla risoluzione di equazioni alle derivate parziali del primo e secondo ordine. Il problema si rappresenta matematicamente con l'equazione

$$F\left(\mathbf{x}, \mathbf{\Phi}, \nabla_{\mathbf{x}}\mathbf{\Phi}, \nabla^2_{\mathbf{x}}\mathbf{\Phi}, \ldots\right) = 0, \quad \mathbf{\Phi} : \mathbf{x} \mapsto \mathbf{\Phi}(\mathbf{x}).$$

Approssimando in maniera efficiente ed efficace derivate prime e seconde, SIREN si comporta molto bene in questi problemi restituendo in output risultati molto affidabili. Per ulteriori approfondimenti, si suggerisce di consultare l'articolo [S].

Confronto tra SIREN e ReLU

Nel caso della rappresentazione implicita, l'obiettivo di una rete neurale è quello di risolvere il problema

$$F\left(\mathbf{x}, \mathbf{\Phi}, \nabla_{\mathbf{x}}\mathbf{\Phi}\right) = 0,$$

dove $\Phi : \mathbb{R}^3 \to \mathbb{R}$ è la SDF e le derivate parziali oltre al primo ordine non appaiono.

In particolare, prende il nome di Eikonal Equation (Equazione Iconale) un'equazione differenziale alle derivate parziali non lineare del primo ordine che descrive la propagazione di fronti d'onda in un mezzo non omogeneo. In termini più semplici, ci dice come un'onda si propaga in uno spazio dove la velocità di propagazione può variare da punto a punto.
L'equazione è solitamente scritta nella forma seguente:

$$\|\nabla\Phi(x)\| = n(x),$$

dove $n(x)$ è l'indice di rifrazione nel punto x, che è inversamente proporzionale alla velocità locale di propagazione.
Il nome deriva dalla parola greca $\epsilon\iota\kappa\omega\nu$ che significa *immagine*. Infatti, questo tipo di equazione appare frequentemente in problemi di ottica geometrica.

La SDF Φ di una superficie S soddisfa il seguente caso particolare di equazione iconale:

$$\|\nabla_{\mathbf{x}}\Phi(\mathbf{x})\| = 1,$$

con condizione al contorno $\Phi = 0$ su S.
La determinazione della SDF ovvero la risoluzione dell'equazione tramite reti neurali è un'alternativa alla risoluzione numerica classica dell'equazione differenziale.
Procedimento: in input viene fornita una nuvola di punti nello spazio tridimensionale e ognuno di questi punti è accoppiato ad una normale, come illustrato in figura 1.1.

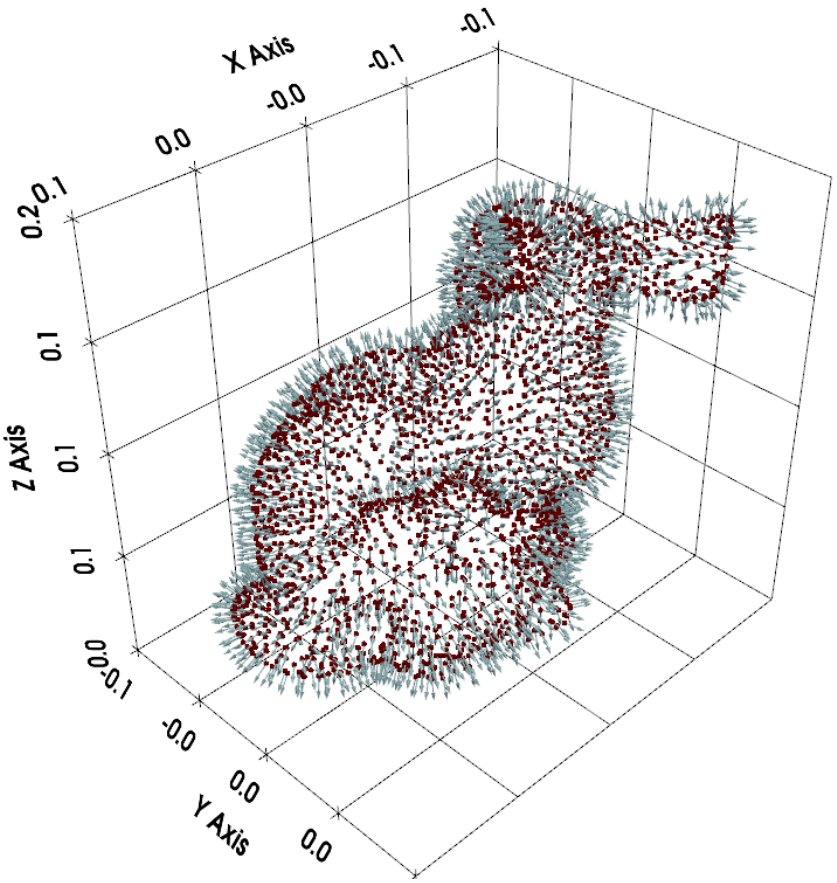

Figura 1.1: *Nuvola di punti associati alle rispettive normali, Coniglio di Stanford.*

Successivamente viene definita la *loss function* come:

$$\mathcal{L}_{\text{sdf}} = \int_{\Omega} \| |\nabla_{\mathbf{x}}\mathbf{\Phi}(\mathbf{x})| - 1 \| \, dx + \int_{\Omega_0} \|\mathbf{\Phi}(\mathbf{x})\| + (1 - \langle \nabla_{\mathbf{x}}\mathbf{\Phi}(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle) \, dx + \int_{\Omega \setminus \Omega_0} \psi\left(\mathbf{\Phi}(\mathbf{x})\right) dx,$$

la quale, detti Ω il dominio di Φ e Ω_0 i punti nella nuvola di input, penalizza: le normali generate con norma lontana da 1 su Ω, le normali generate con una direzione diversa da quella originale su Ω_0 e i punti non sulla superficie con SDF vicina a 0 tramite la funzione $\psi(\mathbf{x}) = \exp(-\alpha \cdot |\mathbf{\Phi}(\mathbf{x})|), \quad \alpha \gg 1$.

Si stabiliscono poi i parametri iniziali, la cui ricerca è uno dei problemi principali [S] e si avvia l'allenamento della rete. La valutazione dei risultati può essere fatta sia tramite l'analisi delle funzioni prodotte in output e quindi usando la loss function, sia tramite la visualizzazione dei risultati. La seconda via è più immediata e intuitiva: una volta generata la mesh usando l'algoritmo *marching cube*[2] sulla SDF approssimata da entrambe le reti neurali, è possibile visualizzare i risultati con una riproduzione della scena di partenza. Le differenze nei risultati sono evidenti, come si vede in figura 1.2, dove la rete che usa funzioni di attivazione ReLU fallisce nel rappresentare i dettagli, mentre SIREN riesce a coglierli molto più accuratamente.

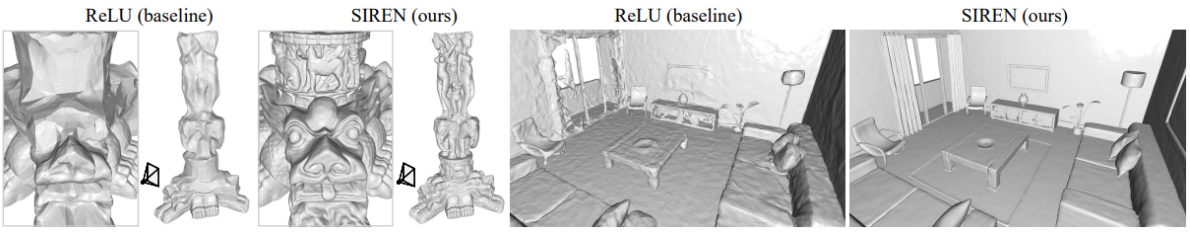

Figura 1.2: *Confronto rappresentazione ReLU e SIREN per una statuetta ed una stanza. Immagine dall'articolo [S].*

[2]Cosa siano mesh e marching cube è spiegato nel capitolo seguente.

Capitolo 2

Visualizzazione di superfici

2.1 Introduzione

Introdotta la rappresentazione di superfici, si può passare alla visualizzazione delle stesse, campo di interesse della computer graphics.
La visualizzazione, o rendering, consiste nella generazione di una scena 2D a partire da un modello 3D, cioè è il processo che permette di passare dalla rappresentazione alla raffigurazione della superficie nella viewport[1]. Il rendering tiene conto di geometrie, illuminazione, texture e colore degli oggetti nella scena. I metodi esistenti sono molteplici e in questa tesi tratteremo solo la pipeline di rendering classica, ossia quello che prevede la creazione di una mesh tramite tassellazione o algoritmo di marching cube. Ciò nonostante, per completezza, viene descritto anche il rendering tramite ray tracing.

Ray tracing

Il ray tracing è una tecnica che permette di rappresentare scene realistiche simulando il comportamento della luce. Il funzionamento (figura 2.1) si basa sul tracciare il percorso di raggi che partono dalla telecamera, attraversano la scena e interagiscono con gli oggetti (riflettendo, rifrangendo o assorbendo). Ogni iterazione determina o perfeziona il colore e l'illuminazione finale dei pixel dell'immagine. La tecnica del ray tracing permette dunque di rappresentare in maniera esatta una superficie, senza passare da una sua approssimazione come invece succede per la pipeline di rendering classica. Per ulteriori dettagli e approfondimenti, si rimanda alla letteratura.

[1] Porzione dell'interfaccia di uno schermo in cui appare la visualizzazione.

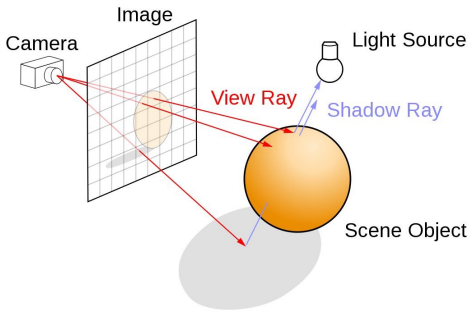

Figura 2.1: *Ray tracing: schema funzionamento.*

2.2 Mesh poligonali

Se si utilizza la pipeline di rendering classica è necessario generare una mesh poligonale. Una mesh poligonale è un oggetto geometrico costruito a partire da blocchi costruttivi di base, detti *primitive grafiche*. Le primitive possono essere punti, linee, curve e superfici. Una mesh consiste di due componenti: una geometrica, che contiene le primitive utilizzate, e una topologica, che definisce la sua struttura a grafo (complesso simpliciale). La geometria si definisce tramite un insieme di vertici sulla superficie $V = (v_1, \ldots, v_n)$, un insieme di facce $F = (f_1, \ldots, f_m)$ dove $f_i \in V \times V \times V$, ed un insieme eventuale di lati. Le facce possono essere quadrilateri o triangoli immersi in $\mathbb{R}^3$ che costituiscono la mesh. Dunque una mesh è un'approssimazione a tratti lineare della superficie.

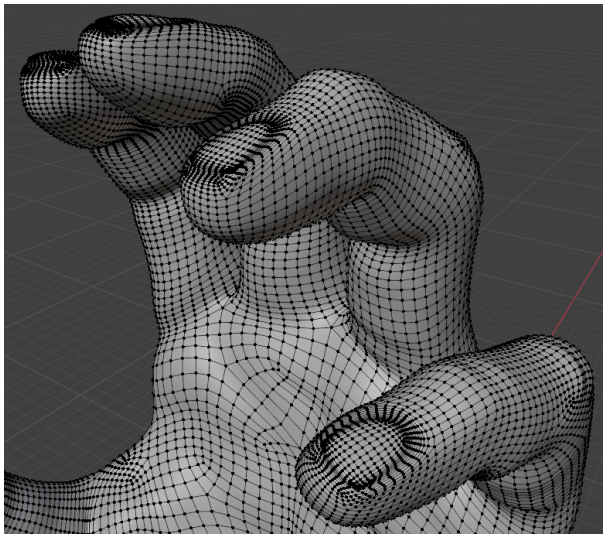

Figura 2.2: *Sezione di mano, mesh poligonale.*

Di una mesh si può calcolare la caratteristica di Eulero, definita come $X = V - E + F$. Siccome la caratteristica di Eulero dipende solo dalla classe di omotopia dell'oggetto considerato, infatti risulta $X = 2(1-g)$ dove g è il genere, è interessante notare che nelle

mesh vale $F \approx 2V$ oppure $E \approx 3V$. Non è quasi mai vera l'uguaglianza perchè possono esserci facce doppie o altri problemi di rendering dovuti alla creazione della superficie.
I formati di file utilizzati nella computer graphics sono molteplici, ma uno che possiamo facilmente interpretare è .obj dove, come si vede nella figura 2.3 (a destra), l'oggetto geometrico è definito a partire da un insieme di vertici e di normali, poi usati per definire le facce.

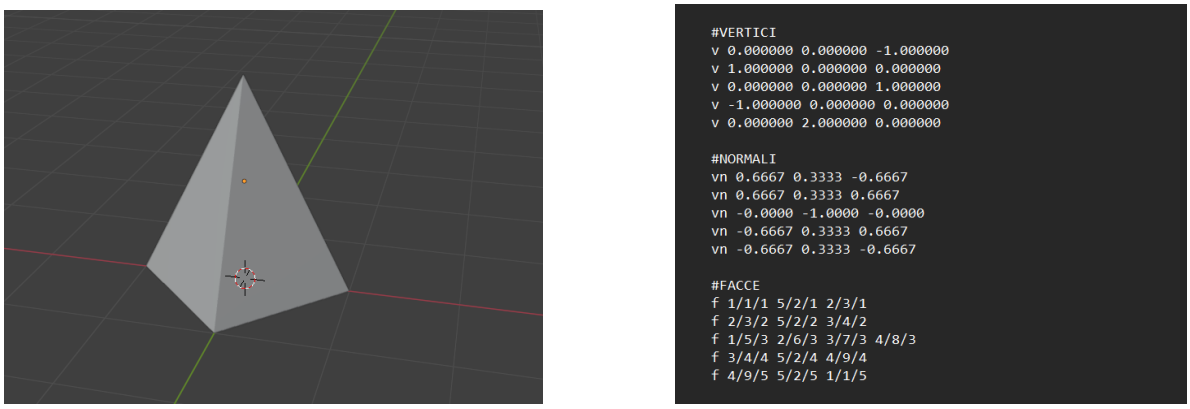

Figura 2.3: *Piramide a 4 facce in Blender (a sinistra) e file OBJ che la definisce (a destra).*

2.3 Visualizzazione da rappresentazione parametrica

Data una rappresentazione parametrica della superficie $S = f(\Omega)$, con $f : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3$, i vertici della mesh si ottengono tassellando il dominio con una griglia e valutando la funzione sui suoi nodi. La tassellazione può essere uniforme o adattiva a seconda della struttura di S: se la curvatura della superficie è costante (o poco variabile) allora una tassellazione uniforme del dominio è sufficiente; quella adattiva risulta utile quando la curvatura della superficie è molto variabile. Infatti, se S presenta tratti quasi piani, la tassellazione può essere molto allargata, viceversa, se esistono regioni ad alta curvatura può essere necessario generare una tassellazione più fitta nelle corrispondenti regioni del dominio al fine di non perdere i dettagli. In figura 2.4 è possibile vedere la differenza tra i due tipi di tassellazione.

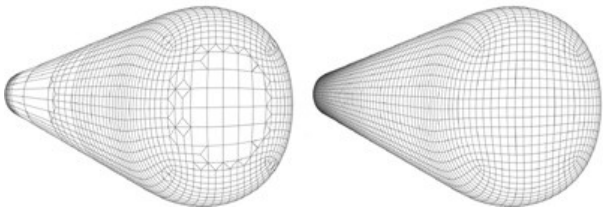

Figura 2.4: *Tassellazione adattiva (sinistra) e uniforme (destra). Immagine dal libro [N].*

2.4 Visualizzazione da rappresentazione implicita

Se la rappresentazione, invece, è implicita, l'algoritmo per generare la mesh è molto più complesso di quello visto per la rappresentazione parametrica. Di algoritmi di questo tipo ne esistono più di uno, ma il principale e più utilizzato è senza dubbio quello chiamato *Marching Cubes*.

Algoritmo Marching Cubes

L'algoritmo suddivide lo spazio tridimensionale in una griglia di cubi. Per ogni cubo valuta il valore delle funzione F su ciascun vertice ed in base al numero di vertici considerati interni o esterni alla superificie, genera una o più facce poligonali. Questi facce vengono poi raccordate tramite algoritmi di *remeshing* per ottenere una superficie il più possibile continua e regolare. Di seguito vengono descritti i passaggi principali dell'algoritmo.

Suddivisione dello spazio: lo spazio tridimensionale viene diviso in una griglia regolare di cubi. Ogni cubo è definito dai suoi 8 vertici, etichettati come $V_0, V_1, V_2, \ldots, V_7$, dove ciascun vertice V_i è descritto dalla sua posizione nello spazio: $V_i = (x_i, y_i, z_i)$.
Classificazione dei vertici: per ogni vertice del cubo, si valuta la funzione implicita $F(x, y, z)$. Se il valore della funzione al vertice V_i è minore o uguale a 0, il vertice è considerato dentro la superficie; altrimenti è considerato fuori. Formalmente:

$$\text{inside}(V_i) = \begin{cases} 1 & \text{se } f(V_i) \leq 0 \\ 0 & \text{se } f(V_i) > 0. \end{cases}$$

Determinazione dell'indice del caso: si assegna un bit (0 o 1) a ciascun vertice del cubo in base alla sua classificazione. Concatenando questi bit si ottiene un indice binario che descrive la configurazione del cubo: $\text{index} = (\text{inside}(V_0), \text{inside}(V_1), \ldots, \text{inside}(V_7))$. Questo indice varia da 0 a 255 e rappresenta una delle possibili combinazioni di vertici dentro e fuori dalla superficie.

Corrispondenza dei facce: l'indice calcolato viene utilizzato per accedere a una tabella di lookup predefinita, che associa ogni possibile configurazione di cubo ad una sezione di mesh. Questa sezione rappresenta un'approssimazione dell'intersezione tra la superficie e il cubo.
Interpolazione dei punti: se $F(V_i) \cdot F(V_j) < 0$, viene generato un vertice lungo il lato tra V_i e V_j tramite interpolazione lineare:

$$P = \frac{V_i \cdot F(V_i) + V_j \cdot F(V_j)}{|F(V_i)| + |F(V_j)|}.$$

Questo permette di aggiornare, usando i valori della funzione, il posizionamento della sezione di mesh generata tramite la tabella.
Costruzione della mesh: alla fine, le facce generate vengono unite per creare una mesh che approssima la superficie implicita originale.

Osservazione 2.1. Le configurazioni sono $255 = 2^8 - 1$ perché entrambe le configurazioni $(1, 1, \ldots, 1)$ e $(0, 0, \ldots, 0)$, cioè "tutti i vertici dentro/fuori", non generano alcuna faccia.

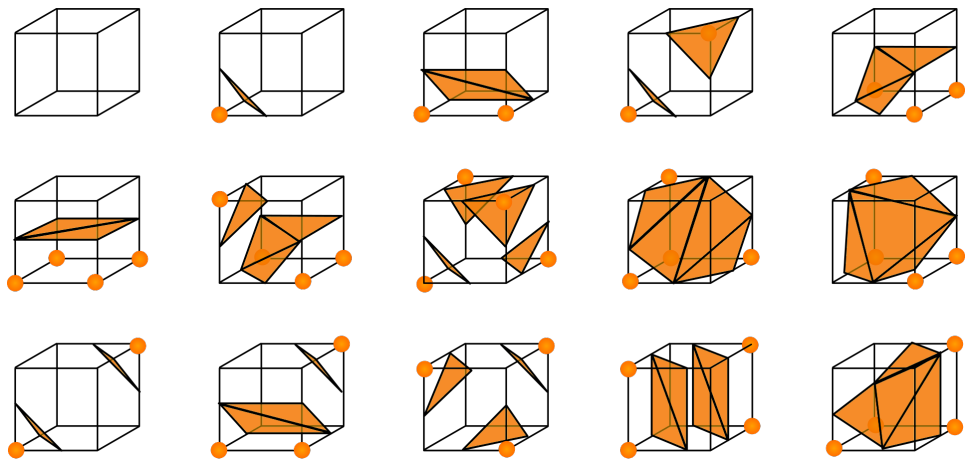

Figura 2.5: *Marching Cube: 15 dei 255 casi possibili.*

2.5 Elaborazione geometrica nelle due rappresentazioni

I problemi in cui si incorre nella modellazione geometrica sono tre: visualizzazione, Point Membership Classification (PMC), manipolazione locale e globale.
La **visualizzazione** è evidente, da quanto detto fin ora, che sia estremamente facilitata

se si usa una rappresentazione esplicita. Infatti, per generare una mesh da una rappresentazione implicita è necessario estrarre l'isosuperficie, mentre nella rappresentazione parametrica è sufficiente valutare la funzione su un campione di punti del dominio.
La **PMC** è il problema che si occupa di determinare se un punto sia interno o esterno alla superficie. La PMC è alla base della computer graphics per via delle sue innumerevoli applicazioni[2]. Per quanto detto sulla rappresentazione implicita, verificare se un punto sia interno, esterno o sulla superficie consiste nel valutare la funzione nel punto stesso. Per quanto riguarda la rappresentazione esplicita, la PMC non è banale.

Esempio 2.2. Riprendendo l'esempio 1.4, per valutare se un punto $P = (x_0, y_0, z_0)$ sia interno al toro nella rappresentazione esplicita è necessario verificare che la distanza dal punto P dalla circonferenza centrata in $(0, 0, 0)$ di raggio R sia minore di r. In altre parole è necessario verificare che

$$\left(\sqrt{x_0^2 + y_0^2} - R\right)^2 + z_0^2 - r^2 < 0.$$

Questa disequazione non è altro che la forma implicita del toro (esempio 1.7) valutata nel punto P. □

Ovviamente la via per risolvere la PMC in modalità esplicita non è solo analitica, ma esistono algoritmi specializzati che utilizzano alcune proprietà geometriche o topologiche degli oggetti rappresentati[3]. In conclusione, nonostante anche la rappresentazione esplicita permetta di risolvere questo problema, quella implicita lo rende estremamente più semplice.
Per quanto riguarda la **manipolazione geometrica**, quale rappresentazione sia più adatta dipende dal tipo di manipolazione richiesto. Se è locale, cioè è necessario applicare modifiche solo su alcuni vertici o facce della mesh, basta proiettarli sul dominio della funzione parametrica e modificare la funzione localmente. Quando le manipolazioni richieste comprendono tutta la superficie, come ad esempio trasformazioni topologiche o operazioni booleane tra superfici, la rappresentazione implicita va privilegiata.

Esempio 2.3. Se si vuole l'unione di due superfici, con la rappresentazione implicita questo si ottiene creando una terza equazione a partire dalle prime due $F(x) = 0$ e $G(x) = 0$, semplicemente moltiplicando $F(x) \cdot G(x) = 0$.

[2]Medical Imaging, studio di collisioni, simulazioni fisiche dei liquidi sono solo alcuni esempi.

[3]Algoritmi di questo tipo sono ad esempio sono: Ray Casting, Point in Polygon test for Meshes oppure Winding Number Algorithm. Per approfondimenti, consultare la letteratura.

2.6 Rappresentazione implicita versus esplicita parametrica

È possibile passare da una rappresentazione esplicita ad una implicita e viceversa, e questo permette di sfruttare i vantaggi di entrambe i metodi. Sebbene la scelta iniziale del tipo di rappresentazione sia guidata dallo scopo per cui viene creata la rappresentazione stessa, non è comunque limitante qualora si dovesse affrontare anche uno degli altri problemi per cui la modalità scelta non sia adatta.

Da parametrica ad implicita: si può fare costruendo la SDF. Il procedimento standard è quello di suddividere lo spazio in una griglia[4] e calcolare la distanza dei nodi dalla faccia della mesh più vicina a loro [S94]. Poiché la SDF è segnata, dato il nodo della griglia $\mathbf{g}$, il punto $\mathbf{c}$ a distanza minima sulla faccia a distanza minima da $\mathbf{g}$, e la normale della faccia $\mathbf{n}(\mathbf{c})$, il segno corrisponde a quello del prodotto scalare $(\mathbf{g} - \mathbf{c})^T \cdot \mathbf{n}(\mathbf{c})$.

Da implicita a parametrica: ci sono più di un algoritmo, ma quello presentato in 2.4 è il più utilizzato.

[4] Uniforme o adattiva.

Capitolo 3

Classi di superfici analizzate

Dopo aver esaminato la teoria di base della rappresentazione di superfici, è ora interessante considerare alcune tecniche precedentemente esposte esplorandone criticità e pregi che le caratterizzano.
Per ogni categoria sono stati presi in considerazione almeno tre oggetti geometrici differenti. Le superfici analitiche sono state scelte in base alla geometria locale e alla curvatura che presentano, poiché questo risulterà utile per la stampa 3D. Per quanto riguarda le superfici a forma libera, sono state scelte in base alla complessità topologica e geometrica, data da spigoli e facce, per analizzare quanto e come questo influisca sul rendering di una scena che le contiene.

3.1 Superfici esplicite parametriche classiche

Sono state scelte cinque superfici analitiche, delle quali è interessante analizzare la funzione generatrice e il dominio su cui essa è definita. Le superfici sono state generate tramite il software Blender usando la funzionalità di *scripting* che permette di lavorare con un codice Python. Il codice:

- discretizza il dominio in una griglia i cui rettangoli hanno dimensioni variabili a seconda delle caratteristiche della superficie, tra cui la curvatura;
- valuta la funzione f sui nodi della griglia generando i vertici della mesh;
- fissa un ordine nei vertici tramite una lista a seconda della corrispondente posizione sulla griglia iniziale;
- genera una mesh tra i vertici usando l'ordine precedentemente fissato.

Esempio 3.1. Questo è un esempio di codice che genera il Nastro di Möbius.
È necessario importare le librerie *bpy, bmesh, numpy, cmath* per far girare il codice.

```
    # Parameters
    u_range = 0.4, v_range = 2*np.pi
    u_step = 32, v_step = 32
    # Function definition
    def f(u, v):
        x = np.cos(v)+u*np.cos(v/2)*np.cos(v)
        y = u*np.sin(v/2)
        z = np.sin(v)+u*np.cos(v/2)*np.sin(v)
        return x, y, z
    # Domain setup
    lu = np.linspace(-u_range,u_range,u_step)
    lv = np.linspace(0,v_range,v_step)
    # Creating mesh in Blender
    bm = bmesh.new()
    # Create vertices
    vertices = [bm.verts.new(f(u,v)) for u in lu for v in lv]
    for i in range(u_step-1):
        for j in range(v_step-1):
            v1 = vertices[i * v_step + j]
            v2 = vertices[i * v_step + (j + 1)]
            v3 = vertices[(i + 1) * v_step + (j + 1)]
            v4 = vertices[(i + 1) * v_step + j]
            bm.faces.new([v1, v2, v3, v4]
    # Create mesh object
    mesh = bpy.data.meshes.new("Moebius")
    bm.to_mesh(mesh)
    bm.free()
    # Add mesh to the scene
    obj = bpy.data.objects.new("Moebius", mesh)
    bpy.context.collection.objects.link(obj)
    bpy.context.view_layer.objects.active = obj
    obj.select_set(True)
```

Nastro di Möbius

La funzione generatrice è

$$f : [-0.4; 0.4] \times [0, 2\pi] \to \mathbb{R}^3, f(u, v) = \begin{pmatrix} \cos(v) + u\cos(v/2)\cos(v) \\ u\sin(v/2) \\ \sin(v) + u\cos(v/2)\sin(v) \end{pmatrix}.$$

La prima variabile del dominio, u, definisce la larghezza del nastro. Il numero di steps è 32 in entrambe le direzioni.

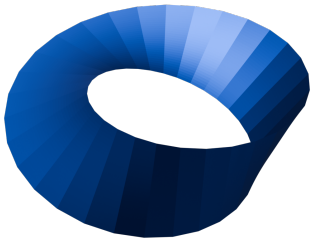

Figura 3.1: *Nastro di Möbius.*

Toro attorcigliato

La funzione generatrice è

$$f : [0; 2\pi] \times [0, 2\pi] \to \mathbb{R}^3, f(u, v) = \begin{pmatrix} \cos(u)(6 - (\frac{5}{4} + \sin(3v))\sin(v - 3u)) \\ (6 - (\frac{5}{4} + \sin(3v))\sin(v - 3u))\sin(u) \\ -\cos(v - 3u)(\frac{5}{4} + \sin(3v)) \end{pmatrix}.$$

Il numero di steps è 128 in entrambe le direzioni.

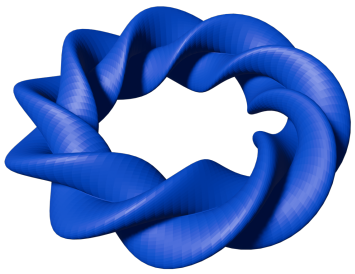

Figura 3.2: *Toro attorcigliato.*

Toro di Clifford

La funzione generatrice è

$$f : [0,\pi] \times [0,2\pi] \to \mathbb{R}^3, f(u,v) = \begin{pmatrix} \frac{\cos(u+v)}{(\sqrt{2}+\cos(v-u))} \\ \frac{\sin(v-u)}{(\sqrt{2}+\cos(v-u))} \\ \frac{\sin(u+v)}{(\sqrt{2}+\cos(v-u))} \end{pmatrix}.$$

Il numero di steps è 8 nella direzione u e 1288 nella direzione v.

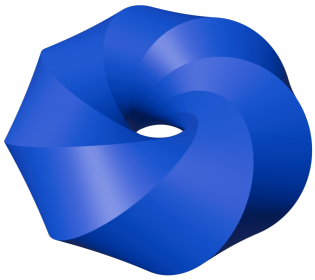

Figura 3.3: *Toro di Clifford.*

Catenoide

La funzione generatrice è

$$f : [-\pi,\pi] \times [-\pi,\pi] \to \mathbb{R}^3, f(u,v) = \begin{pmatrix} 2\cosh(\frac{v}{2})\cos(u) \\ v \\ 2\cosh(\frac{v}{2})\sin(u) \end{pmatrix}.$$

Il numero di steps è 32 nella direzione u e 128 nella direzione v.

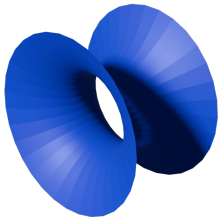

Figura 3.4: *Catenoide.*

Superficie di Dini

La funzione generatrice è

$$f : [0, 4\pi] \times [0, 2] \to \mathbb{R}^3, f(u, v) = \begin{pmatrix} \cos(u)\sin(v) \\ \sin(u)\sin(v) \\ (\cos(v) + \log(\tan(\frac{v}{2}) + 10^{-2})) + \frac{u}{5} \end{pmatrix}.$$

Il numero di steps è 128 nella direzione u e 128 nella direzione v.

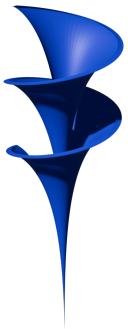

Figura 3.5: *Superficie di Dini.*

3.2 Superfici esplicite parametriche a forma libera

In generale, con superficie a forma libera si intende una superficie non esprimibile tramite assemblaggio di superfici analitiche note, per cui è dunque necessario utilizzare altre tecniche che la definiscono mediante punti. I punti possono essere di controllo se si usano strumenti come le superfici spline o le patch di Bézier, oppure possono costituire nuvole di punti, ottenute tramite scansione 3D, dove vengono interpolati per ricostruire la superficie iniziale. Un esempio in questo senso è la ricostruzione dello Stanford Bunny tramite la rete neurale SIREN (3.4).
Una delle possibilità per rappresentare superfici parametriche a forma libera è quella di usare le NURBS. In particolare, di seguito verranno analizzate sia superfici con particolari simmetrie, sia superfici che necessitano di più di una superficie spline per essere rappresentate. L'obiettivo, in entrambi i casi, è quello di ridurre al minimo i punti di controllo e il numero di spline utilizzate.
Su Blender è possibile generare spline in più modi. Il più immediato è quello che utilizza l'interfaccia, permettendo di modificare la posizione dei punti di controllo direttamente col mouse. Ho trovato però più interessante utilizzare ancora una volta la sezione di *scripting*: questo sistema offre la possibilità di controllare in modo assolutamente libero

posizione, numero dei punti di controllo e grado delle spline, così da ottenere curve e superfici spline esattamente come gradito. Di seguito la porzione di codice commentato che permette di generare una spline definendone grado e punti di controllo (è necessario importare il pacchetto *bpy*).

```
def create_NURBS_curve_with_weights():
    # Define control points and their weights for the NURBS curve
    control_points = [
        ( 0,0,0,1), (0, 4,0,1), ( 0, 2,2,1), ( 0,2,4,1),
        ( 0,1,6,1), ( 0,3,8,1)
    ]
    # Create a new curve object
    bpy.ops.curve.primitive_nurbs_curve_add(location=(0, 0, 0))
    curve_obj = bpy.context.object
    curve_obj.name = 'NURBSCurveWithWeights'
    # Access the first spline
    spline = curve_obj.data.splines[0]
    # Ensure it has exactly 4x4 control points (16 points)
    while len(spline.points) < len(control_points):
        spline.points.add(1)
    while len(spline.points) > len(control_points):
        spline.points.remove(len(spline.points) - 1)
    # Assign the control points and their weights
    for i, (x, y, z, w) in enumerate(control_points):
        spline.points[i].co = (x, y, z, w)
    # Set the order and endpoint usage
    spline.order_u = 4
    spline.use_endpoint_u = True

create_NURBS_curve_with_weights()
```

3.2.1 Superfici cross-sectional

Nella modellazione geometrica può essere complesso generare e manipolare superfici mantenendole regolari. Per questo risulta utile il concetto di superfici cross-sectional, ossia generate da curve, dette di profilo. Le superfici generate in questo modo possono essere modellate agendo solo sulle curve di profilo, permettendo di mantenere regolarità

e uniformità. Di seguito ne vengono introdotte tre: di estrusione, di rotazione e di swinging.

Superfici di estrusione

Una superficie spline ottenuta per estrusione è generata a partire da una curva spline di profilo

$$C(u) = \sum_{i=1}^{n+K+1} P_i N_{i,n}(u),$$

che poi viene estrusa verso un'unica direzione $\mathbf{W}$. La spline finale ha formula

$$S(u,v) = \sum_{i=1}^{n+K+1} \sum_{j=1}^{2} P_{ij} N_i(u) N_j(v),$$

in cui fissando il parametro $\tilde{u}$ si ottiene il segmento retto da $C(\tilde{u})$ a $C(\tilde{u}) + dW$, mentre fissando il parametro $\tilde{v}$ si ottiene la spline di profilo traslata di $\tilde{v}dW$, ossia $S(u,\tilde{v}) = C(u) + \tilde{v}dW = \sum_{i=1}^{n+K+1}(P_i + \tilde{v}dW)N_{i,n}(u)$.
Nel caso analizzato, è stata costruita una curva spline chiusa di grado 3 con punti di controllo: $[(-1,0,0,1),(-2,1,0,1),(2,1,0,1),(1,0,0,1),(1,0,0,1),(2,-1,0,1),(-2,-1,0,1),(-1,0,0,1),(-1,0,0,1),(-2,1,0,1),(2,1,0,1),(1,0,0,1)]$.

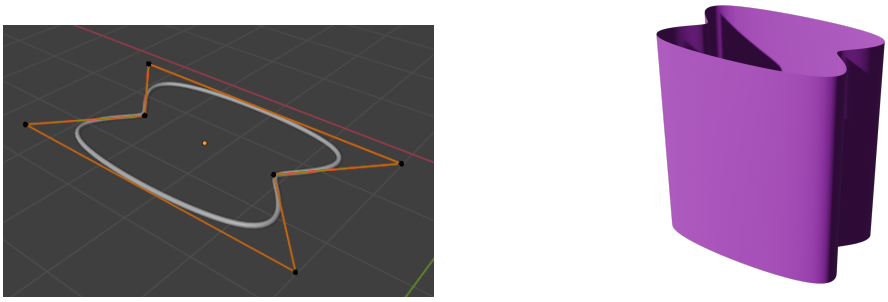

Figura 3.6: *Superficie spline ottenuta per estrusione (destra) e curva profilo con punti di controllo (sinistra).*

Superfici di rotazione

Una superficie spline può anche essere generata utilizzando le sue simmetrie. In questo caso, poiché il vaso ha una simmetria di rotazione rispetto all'asse z, è sufficiente definire una spline di profilo nel piano yz e poi ruotarla intorno all'asse z seguendo il profilo di una circonferenza. È necessario notare che il cerchio può essere ottenuto solo tramite curve NURBS.

Data una curva di profilo

$$C(v) = \sum_{j=1}^{m+H+1} P_j R_j^m(v),$$

considerata la curva NURBS che genera il cerchio con partizione nodale

$$\Delta_u^* = \left\{0, 0, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}, \frac{3}{4}, 1, 1, 1\right\}$$

e pesi

$$W = \left\{1, \frac{\sqrt{2}}{2}, 1, \frac{\sqrt{2}}{2}, 1, \frac{\sqrt{2}}{2}, 1\right\},$$

possiamo ottenere la superficie NURBS di rotazione con

$$S(u,v) = \frac{\sum_{i=1}^{9}\sum_{j=1}^{m+H+1} w_{ij} P_{ij} N_{i,3}(u) N_{j,m}(v)}{\sum_{i=1}^{9}\sum_{j=1}^{m+H+1} w_{ij} N_{i,3}(u) N_{j,m}(v)}.$$

I pesi w_{ij} sono il prodotto dei pesi w_i e w_j. I punti di controllo vengono presi dalla curva di profilo e dal cerchio NURBS secondo la regola

$$P_{ij} = \begin{cases} P_j & \text{se } i = 1, \\ \text{Punti di controllo del cerchio NURBS} & \text{se } i = 2, 3, \ldots, 9 \end{cases}.$$

La spline utilizzata per la curva di profilo è di grado 3 con punti di controllo

$$[(0,0,0,1), (0,4,0,1), (0,2,2,1), (0,1.5,4,1), (0,1,6,1), (0,3,8,1)]$$

di cui interpola quello iniziale e quello finale.

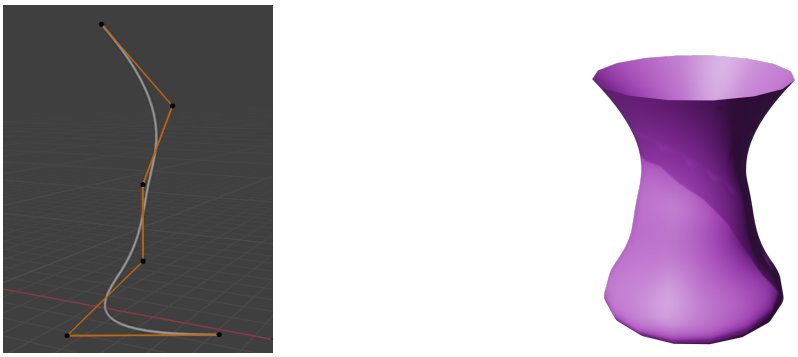

Figura 3.7: *Superficie spline ottenuta per rotazione (destra) e curva profilo con punti di controllo (sinistra).*

Superfici di swinging

Una superficie di swinging è generata a partire da due curve: una di profilo e una di traiettoria. Partendo da due curve di equazioni rispettivamente

$$P(u) = \sum_{i=1}^{n+H+1} P_i R_i^n(u)$$

e

$$T(v) = \sum_{j=1}^{m+K+1} T_j R_j^m(v),$$

si ricava la superficie

$$S(u,v) = (\alpha P_x(u) T_x(v), \alpha P_x(u) T_y(v), P_z(u))^T .$$

α è un valore arbitrario ed è interessante notare come, fissando il parametro $\tilde{u}$ si ottiene la curva $T(v)$ scalata opportunamente e, invece, fissando $\tilde{v}$ si ottiene la curva $P(u)$ scalata opportunamente. Nel caso analizzato è stata generata una spline di profilo di grado 3 con punti base:

$$[(0,0,0,1),(0,4,0,1),(0,2,2,1),(0,2,4,1),(0,1,6,1),(0,3,8,1)].$$

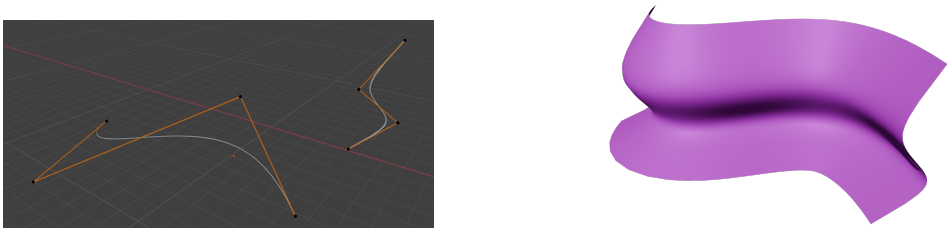

Figura 3.8: *Superficie spline ottenuta per swinging (destra) e curva profilo con punti di controllo (sinistra).*

3.2.2 Superfici esplicite con più domini parametrici

Un'altra tecnica per rappresentare le superfici a forma libera è quella di usare più parametrizzazioni, ciascuna delle quali rappresenta una patch della superficie. Un esempio fornito nel seguito è quello della *Utah teapot*.

La **Utah Teapot** è una superficie a patch, ovvero è definita dall'unione di 32 patch di Bézier (e relativi domini parametrici distinti). La Utah Teapot è una superficie "storica" che ha assunto nel corso degli anni il ruolo di superficie di riferimento per testare gli algoritmi di rendering. È una superficie relativamente problematica poiché presenta

complessità geometriche, grazie al beccuccio e al manico che si inseriscono nel corpo della teiera, ma anche di rendering essendo una superficie non molto regolare. Queste caratteristiche fanno della teapot uno strumento utile per gli sviluppatori che, utilizzando tutti lo stesso riferimento, possono confrontare realisticamente le differenze tra i vari algoritmi di rendering.

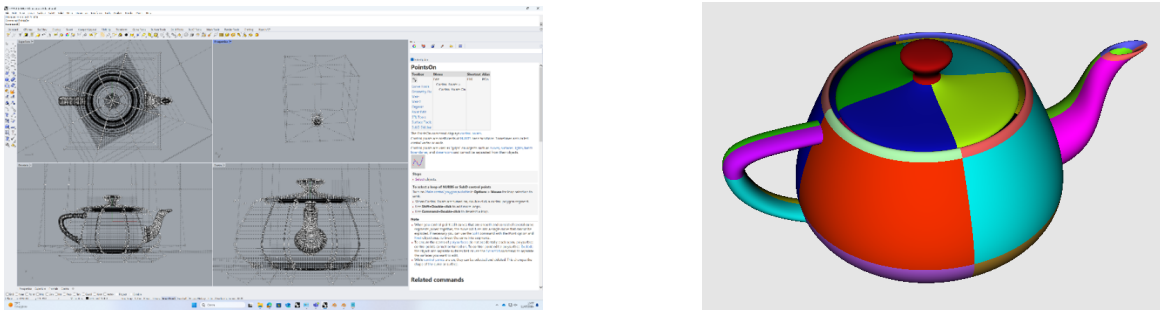

Figura 3.9: *Teapot: visualizzazione tramite Rhino dei punti di controllo (sinistra) e visualizzazione delle 32 patch di Bézier (destra).*

3.3 Superfici esplicite rappresentate da mesh poligonali

Un altro modo di rappresentare le superfici parametriche a forma libera è quello di costurire direttamente la mesh poligonale. In questo caso si ha massima libertà nel posizionamento dei vertici e nella definizione delle facce, ma la superficie risulterà lineare a tratti, quindi non liscia già nella rappresentazione. Nella maggior parte dei casi, questo tipo di rappresentazione deriva da una scansione 3D che ha campionato una nuvola di punti.

Le superfici di questo tipo permettono anche di esplorare alcune difficoltà della stampa 3D di oggetti che presentano irregolarità, non apprezzabili usando solo superfici esplicite note, ma questo verrà discusso nel capitolo 5.

Di seguito alcuni esempi scaricati dalla Stanford 3D Scanning Repository [S3D]. Nel drago e nella statuetta possiamo vedere che l'osservazione sulla formula di Eulero viene rispettata. Per quanto riguarda la mano, ciò non è vero: spesso nelle superfici ottenute per scansione 3D vengono generati vertici o facce aggiuntive rispetto a quelle che dovrebbero esserci.

	N. vertici	N. lati	N. facce	Rendering
Mano	19.986	39.968	19.986	
Drago	3.609.455	10.828.359	7.218.906	
Statuetta	4.999.996	15.000.000	10.000.000	

Tabella 3.1: Dati delle mesh poligonali e relativo rendering effettuato con Blender.

3.4 Superficie implicita SDF: SIREN

Il codice utilizzato per implementare SIREN può essere riassunto nei seguenti passaggi:

- Upload dei dati: viene letto il file di input che consiste in una nuvola di punti in formato .xyz;
- Organizzazione dei dati iniziali in un dataset TensorFlow[1];
- Loop di allenamento;
- Test e valutazione del modello: la rete addestrata viene testata su un volume contenente la superficie attraverso una griglia 3D regolare per ottenere la SDF. Poi viene estratta e visualizzata, tramite l'algoritmo marching cube, l'isosuperficie della SDF, ricostruendo lo Stanford Bunny.

Di seguito possono essere osservati la nuvola di punti di partenza e il risultato ottenuto. Oltre ai punti e alle normali ricevute in input, al fine di migliorare le proprietà di approssimazione della rete, vengono aggiunti ulteriori punti e le rispettive normali: una visualizzazione di questa aggiunta è possibile in figura 3.10.

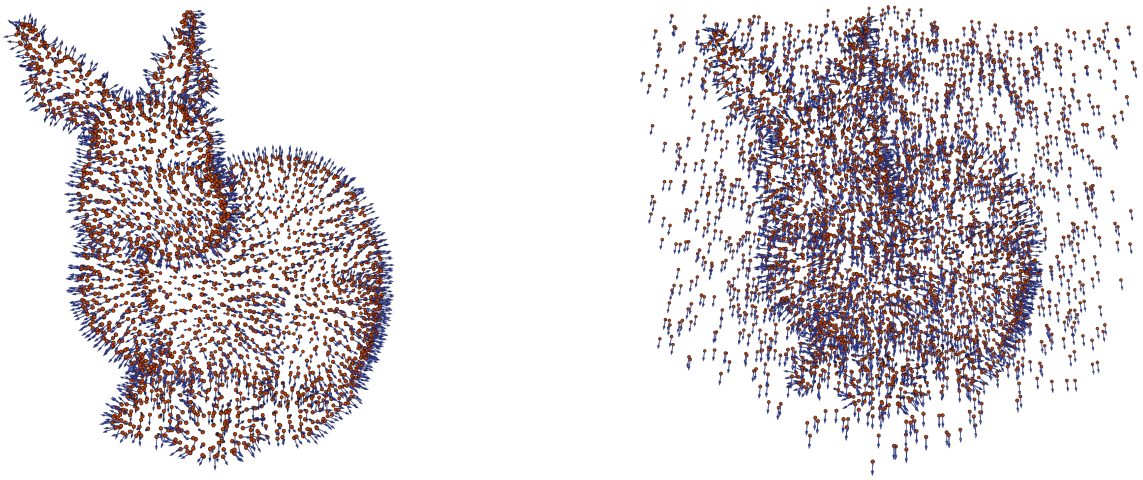

Figura 3.10: *Nuvola di punti e normali associate di partenza (sinistra) e con aggiunte (destra).*

[1]Particolare tipo di dataset, sviluppato da Google, che permette di organizzare e ottimizzare i dati per l'allenamento di una rete neurale.

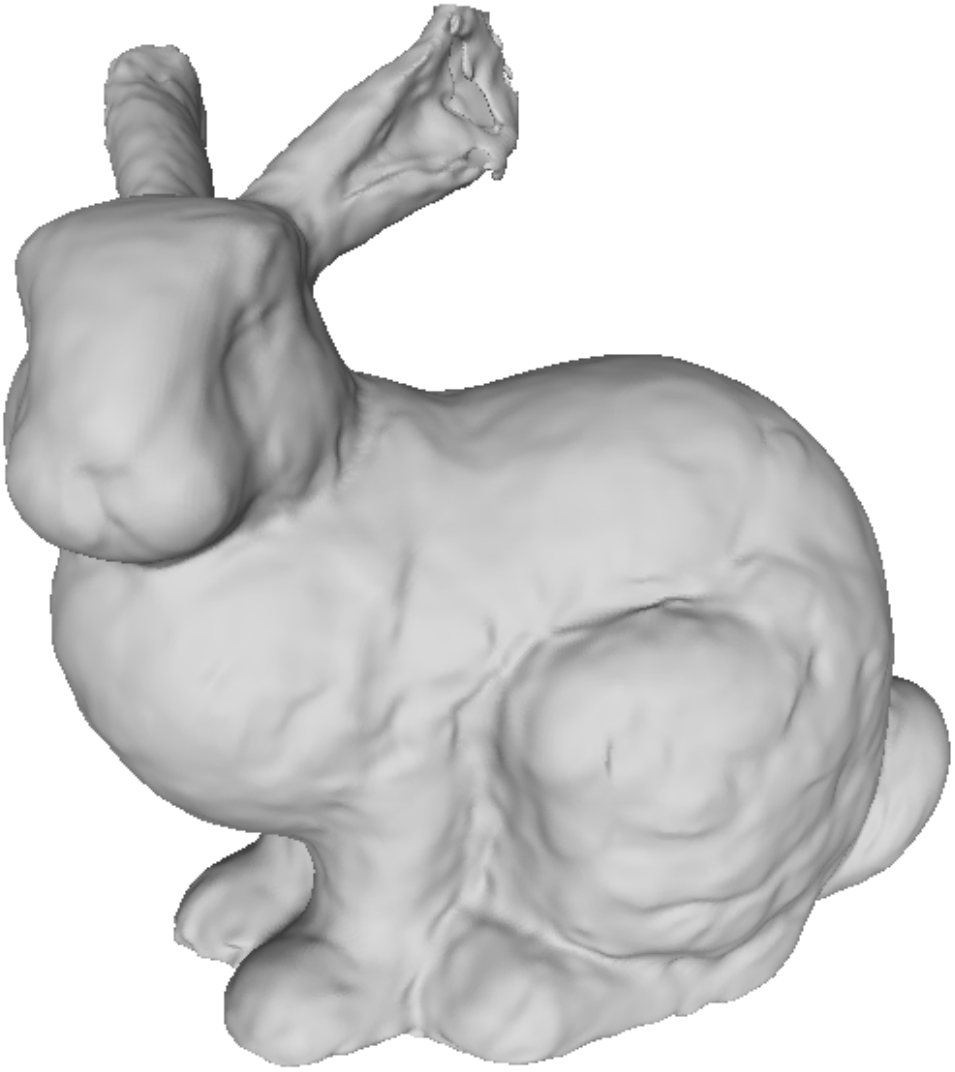

Figura 3.11: *Superficie prodotta in output.*

Capitolo 4

Navigazione virtuale della scena 3D

In questo capitolo si descrive l'esperienza di navigazione virtuale di una scena 3D che rappresenta un museo virtuale che espone le superfici costruite e descritte in precedenza. Oltre alla finalità espositiva, generare il museo virtuale ha permesso di esplorare la navigazione virtuale in Blender.

Figura 4.1: *Vista interna del museo virtuale.*

Il museo è stato modellato con il software Blender a partire da una scena vuota in cui sono state aggiunte le stanze, i supporti, i faretti, e le superfici. Come ultimo passaggio prima della navigazione virtuale si sono definite le luci, i materiali e le texture delle componenti della scena. La scena virtuale può essere esplorata in più modi su Blender, uno tra questi è quello di camminare al suo interno tramite la modalità *Walk Navigation*.

La navigazione può essere fatta utilizzando l'interfaccia di Blender, oppure utilizzando un caschetto per la realtà virtuale, altrimenti detto Visore 3D. Un visore 3D utilizza sensori e display per permettere a chi lo indossa di immergersi in un mondo totalmente virtuale (realtà virtuale) oppure in una realtà che contiene oggetti virtuali (realtà aumentata). Le caratteristiche principali di un visore 3D sono:

- Display Binoculare: ogni occhio del visore è dotato di un display separato (spesso OLED o LCD), che visualizza immagini stereoscopiche. La visione stereoscopica è quella che crea l'effetto di profondità e lo fa mediante la differenza di immagini tra i due occhi.
- Tracciamento della Testa: sensori come accelerometri, giroscopi e magnetometri monitorano il movimento della testa. Questi dati sono usati per aggiornare la visualizzazione in tempo reale e mantenere la prospettiva corretta.
- Rilevamento dell'Interazione: sensori di posizione e telecamere integrati possono tracciare i controller o le mani dell'utente, permettendo interazioni precise con l'ambiente virtuale.
- Rendering: un'unità di elaborazione (come una GPU) calcola e rende i fotogrammi 3D che vengono visualizzati sui display, aggiornando rapidamente le immagini in base ai movimenti dell'utente.

Queste componenti lavorano insieme per creare un ambiente virtuale immersivo che si adatta ai movimenti e alle interazioni dell'utente.
A titolo illustrativo, con l'aiuto del tutor che mi ha affiancato, ho creato anche un video che permettesse di visualizzare il museo virtuale lungo una camminata che lo attraversa. Questo video è stato creato definendo il percorso della videocamera attraverso la scena e generando un frame ogni 1/24 di secondo, così da produrre una sequenza di frame che potesse realisticamente simulare il movimento della videocamera lungo il cammino. Il risultato è disponibile su Youtube tramite il qr-code:

Il video è a 360°, cioè permette di muovere la videocamera durante la passeggiata. Se visualizzato tramite un dispositivo tradizionale come uno smartphone o un computer, sarà possibile muoverla con il mouse o con le freccette; in alternativa, indossando un

visore, sembrerà di muoversi dentro il museo e, semplicemente ruotando la testa, sarà possibile aggirarsi all'interno della scena.

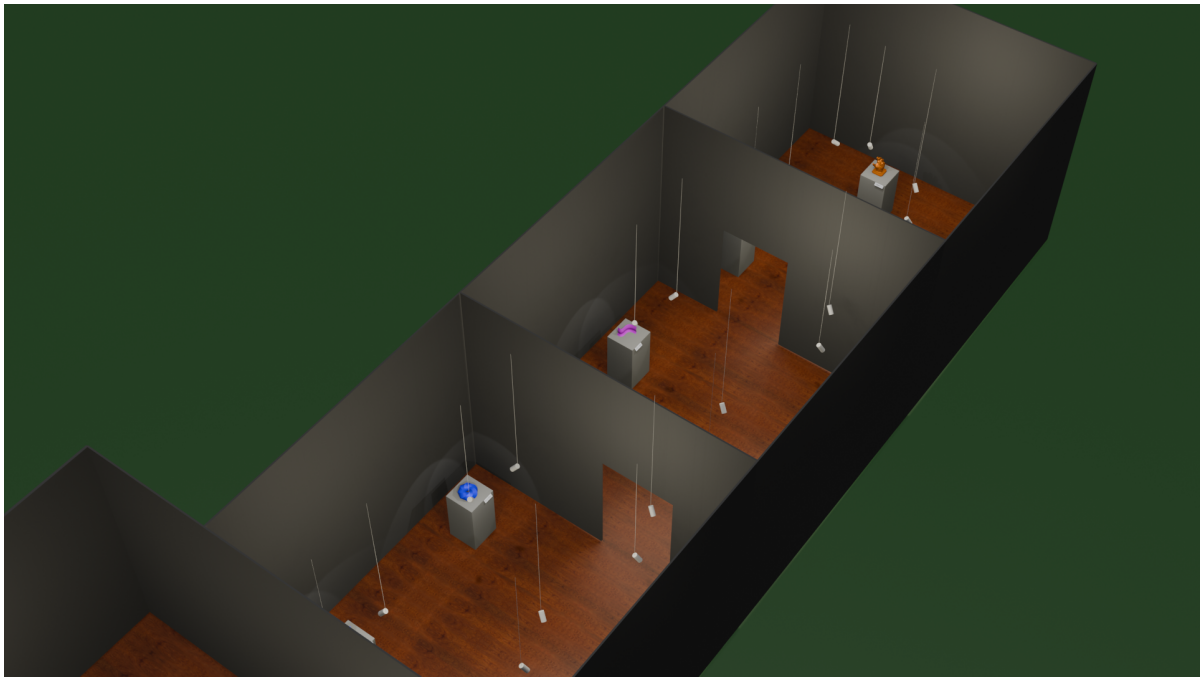

Figura 4.2: *Vista senza soffitto del museo virtuale.*

Figura 4.3: *Vista senza pareti del museo virtuale.*

Capitolo 5

Stampa 3D di superfici selezionate

5.1 Panoramica sui metodi di produzione

I metodi con cui un oggetto può essere costruito a livello industriale sono tre: formativo, sottrattivo e additivo. Il primo utilizza uno stampo che deforma un materiale, generalmente caldo, per modellarlo nella forma desiderata. Il metodo sottrattivo, invece, parte da un blocco più grande dell'oggetto da costruire e procede per rimozione di materiale: è paragonabile alla scultura. Il metodo additivo, infine, consiste nell'aggiungere materiale strato per strato fino al completamento della costruzione dell'oggetto.
Ognuna di queste tecniche presenta vantaggi e svantaggi, economici e tempistici; la scelta viene effettuata a seconda dell'obiettivo in termini di precisione e numero di prodotti richiesti. Il metodo formativo è, intuitivamente, molto vantaggioso per la produzione in serie, poiché il grande costo iniziale della produzione dello stampo viene poi ammortizzato dal basso costo di produzione dei singoli oggetti. La tecnica sottrattiva permette un'eccellente precisione ma produce un grande spreco di materiali e costo in termini di tempo. L'ultima tecnica, quella additiva ossia la stampa 3D, è quella di cui parleremo più approfonditamente in questo capitolo. Questo metodo è ottimale per la produzione di prototipi in quando la stampa è molto veloce e sostanzialmente non necessita di alcun investimento iniziale escludendo quello del materiale. Basso costo e velocità, ovviamente, sono relativi agli altri due metodi. Un altro vantaggio è la libertà geometrica concessa sui modelli da stampare poiché non vi è quasi alcuna limitazione in questo senso, mentre per la tecnica sottattiva spesso i vincoli meccanici dei bracci robotici impongono un ri-orientamento del blocco di lavoro.

D'ora in poi, la discussione verterà solamente sul processo di stampa 3D poiché gli altri due sono fuori dagli scopi di questa trattazione.

5.2 Stampa 3D

La stampa 3D ha molteplici tecnologie che sono tra loro completamente diverse per quanto riguarda i materiali utilizzati e le tecniche di costruzione degli strati, ma tutte condividono i seguenti tre passaggi fondamentali.

5.2.1 Processo di stampa 3D

File 3D

Un file 3D è un modello digitale in tre dimensioni che può essere generato tramite un modello CAD (*Computer Aided Design*) oppure tramite una scansione 3D. La tecnica più utilizzata e sicuramente più precisa è comunque la prima.
I software di CAD sono molti e ognuno è specializzato in qualche tipo di realizzazione geometrica. Per questa tesi abbiamo deciso di utilizzare Blender poiché ammette molte funzionalità anche in campo matematico permettendo, ad esempio, di generare superfici con relativa facilità tramite una loro parametrizzazione. Questo software pone alcune limitazioni quando si vuole lavorare con curve o superfici spline poiché, quasi sempre, per poter applicare delle trasformazioni alle superfici è necessario convertirle in mesh perdendo così la struttura di spline e, di conseguenza, la facilità e la regolarità con cui è possibile applicare trasformazioni. Per questo motivo molti esperti del settore utilizzano altri software come, ad esempio, Rhinoceros 3D. Per esplorare strutture più complesse di superficie spline, come la *Teapot a patches*, abbiamo utilizzato anche quest'ultimo software, ma limitatamente alla visione della struttura dei punti di controllo e delle patches.

Conversione in file OBJ o STL

Ogni software CAD (o scanner 3D) lavora con formati di file differenti, ma ai fini della stampa 3D è poi necessario convertirli in un formato leggibile dai software delle stampanti. Per questo si esportano i file in formato STereoLithography (STL), Wavefront OBJ o altri simili. La struttura di un file OBJ può essere vista nell'immagine 2.3 a destra.
Una volta che è stato generato il file OBJ, questo viene letto da un software chiamato *slicer* che lo converte poi in G-code, un linguaggio di programmazione che descrive percorso, velocità e altre caratteristiche che devono essere seguiti dalla stampante.
Lo slicer permette di regolare molti parametri della stampa, alcuni dei quali sono velocità dei movimenti di stampa, altezza strato, temperatura strato e riempimento, che indica la percentuale di materiale da inserire dentro la superficie, che varia a seconda

della funzione dell'oggetto: se è solo espositiva è sufficiente rimanere sotto il 20%, per altri scopi si può alzare anche fino al 100%. Ovviamente più è pieno l'oggetto, maggiori sono il tempo e il costo di stampa.
Lo slicer è in grado di generare, ove servano, i supporti: strutture di sostegno all'oggetto da stampare per evitare difetti di stampa discussi in seguito.

Stampa e rifinitura

L'ultimo passaggio è quello della stampa, dove basta caricare il file nella stampate. Qualora siano stati inseriti dei supporti, è poi necessario rifinire l'oggetto rimuovendoli e pulendo la superficie da eventuali imperfezioni.
Per stampe più complesse e industriali sono poi necessarie accortezze superiori in fase di rifinitura e di rimozione dell'oggetto dal piano di lavoro, ma nel nostro caso questo non è necessario.

5.2.2 Tecniche di stampa 3D

I materiali con cui vengono effettuate le stampe sono principalmente due: polimeri e metalli. I metalli sono generalmente sotto forma di polvere che viene poi fusa tramite laser o unita tramite legante a generare la forma richiesta. I polimeri si suddividono in *termoplastici* e *termoindurenti*: i primi vengono fusi e poi depositati, mentre i secondi partono da uno stato liquido per poi essere induriti tramite esposizione a raggi UV o altri raggi di luce.
Le tecniche principali di stampa 3D sono sette:

- Getto di materiale;
- Fusione a letto di polvere;
- Fotopolimerizzazione a vasca (o Polimerizzazione in vasca);
- Estrusione di materiale;
- Getto di legante;
- Deposizione diretta di energia;
- Laminazione a foglio.

L'unica che verrà approfondita e sperimentata è quella dell'estrusione di materiale. La tecnologia impiegata in questa tecnica è detta Fused Filament Fabrication (FFF) oppure Fused Deposition Modeling (FDM): viene posta una bobina di filamento sulla stampante che, inserito nel beccuccio estrusore viene fuso e depositato sugli strati precedenti della stampa.

5.2.3 Problematiche di stampa

I problemi in cui è più facile incorrere sono dovuti al raffreddamento del materiale. I fattori che possono influire su questo parametro sono la velocità di stampa, la temperatura di stampa, la presenza di un piano riscaldato e la geometria dell'oggetto.
Il raffreddamento dell'oggetto influisce poi anche sulle proprietà strutturali dello stesso, dunque quando si stampa un prototipo, magari piccolo, di un oggetto che si vorrà creare poi più grande e il cui utilizzo non sia limitato a quello espositivo, è necessario tener conto anche di eventuali criticità strutturali non presenti nel prototipo.

Warping

Uno dei problemi più facilmente riscontrati è il *warping*: distorsione o deformazione di un materiale, solitamente visibile come piegamenti, arricciamenti o sollevamenti degli angoli o dei bordi dovuto ad un raffreddamento non uniforme oppure a stress termici.
Questo problema viene parzialmente risolto, almeno per gli strati iniziali, con l'introduzione del piano di stampa riscaldato. Nelle fasi iniziali di stampa spesso viene inserito un piano su cui poi stampare l'oggetto che, per evitare che si pieghi, deve raffreddarsi uniformemente. Tutto ciò è possibile solo grazie al piano riscaldato.

Esempio 5.1. Un tentativo di stampa del toro attorcigliato, figura 3.3, mostra molto chiaramente questo effetto. La velocità di stampa pre-impostata è di 60mm/s; per questa stampa è stata utilizzata una velocità di 100mm/s. La velocità eccessiva che è stata selezionata non ha fornito il tempo sufficiente agli strati inferiori di raffreddarsi, permettendo quindi alla stampante di depositare materiale su uno strato ancora caldo. Per via della curvatura dell'oggetto, nella fase iniziale viene depositato lo strato leggermente fuori dalla base data da quello precedente. Siccome la base è ancora calda, il raffreddamento dell'ultimo strato avviene prima nell'estremo a sinistra rispetto al resto, che quindi si curva poiché sottoposto a stress termico. La stessa superficie è poi stata stampata ad una velocità di 50mm/s ma anche questo non ha funzionato: il tempo tra l'inizio e la fine della deposizione dello strato a questo punto diventa troppo e il warping succede per il motivo opposto, cioè la prima parte di strato si raffredda quando ancora l'ultima deve essere stampata.

Stampa con angolo minore di 45°

Tutte le volte che è necessario stampare una superficie con un angolo dal piano minore di 45°, è anche necessario impostare i supporti. Qual'ora questo non venisse fatto, potreb-

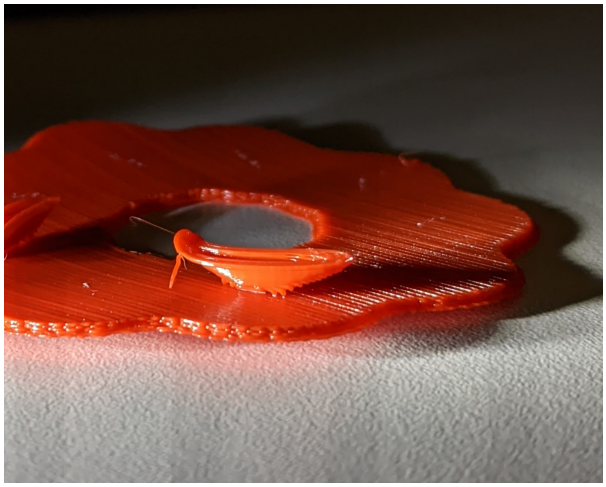

Figura 5.1: *Tentativo di stampa: toro attorcigliato. Warping del lato sinistro.*

bero nascere problemi legati all'assenza di una base su cui stampare lo strato successivo. In questo senso sono stati fatti due esperimenti: il toro attorcigliato e un vaso.

Esempio 5.2. Se si guarda più attentamente l'immagine sopra, si può anche notare che, sul lato destro che non presenta warping, invece è presente del materiale colato, proprio dovuto alla mancanza di un sostegno su cui stampare gli strati successivi. □

Figura 5.2: *Tentativi di stampa: toro attorcigliato, colatura.*

Meno percepibile ma comunque presente, la colatura è successa anche nella stampa di un vaso. Questo vaso presenta anche warping che è dovuto alla sua geometria.

Riduzione dimensione buchi

La dimensione reale, ossia quella nell'oggetto stampato, di buchi o fori è quasi sempre diversa da quella teorica. Questo è dovuto alla pressione che il beccuccio esercita sullo strato che sta depositando, finalizzata ad una migliore adesione dello stesso, ma che causa un allargamento della larghezza teorica dello strato che viene depositato. Se dovessimo stampare dei buchi la cui dimensione è importante sia quella giusta, in fase di progettazione è necessario tenere conto anche di questo rimpicciolimento.

Figura 5.3: *Tentativo di stampa: vaso, colatura.*

5.2.4 Sperimentazione

Tenendo conto di tutte le criticità elencate in precedenza, sono state eseguite le seguenti stampe, con l'aiuto di supporti ove servissero: una mano (forma libera) e un toro attorcigliato. La prima indaga l'effetto dei supporti in una struttura con angoli vistosamente minori di 45°, mentre la seconda esplora quanto i supporti riescano a correggere gli errori di stampa incontrati nei test iniziali.
Ovviamente stampare con i supporti implica che la finitura della superficie rimanga più grezza perché la loro rimozione lascia residui che non possono essere totalmente puliti. Qualora servisse produrre oggetti con supporti con una finitura ecellente, sarebbe necessario utilizzare un altro tipo di stampante che permetta di stampare in due materiali, così da poter generare i supporti in materiale idrosolubile al fine di non lasciare residui sulla superficie finale.

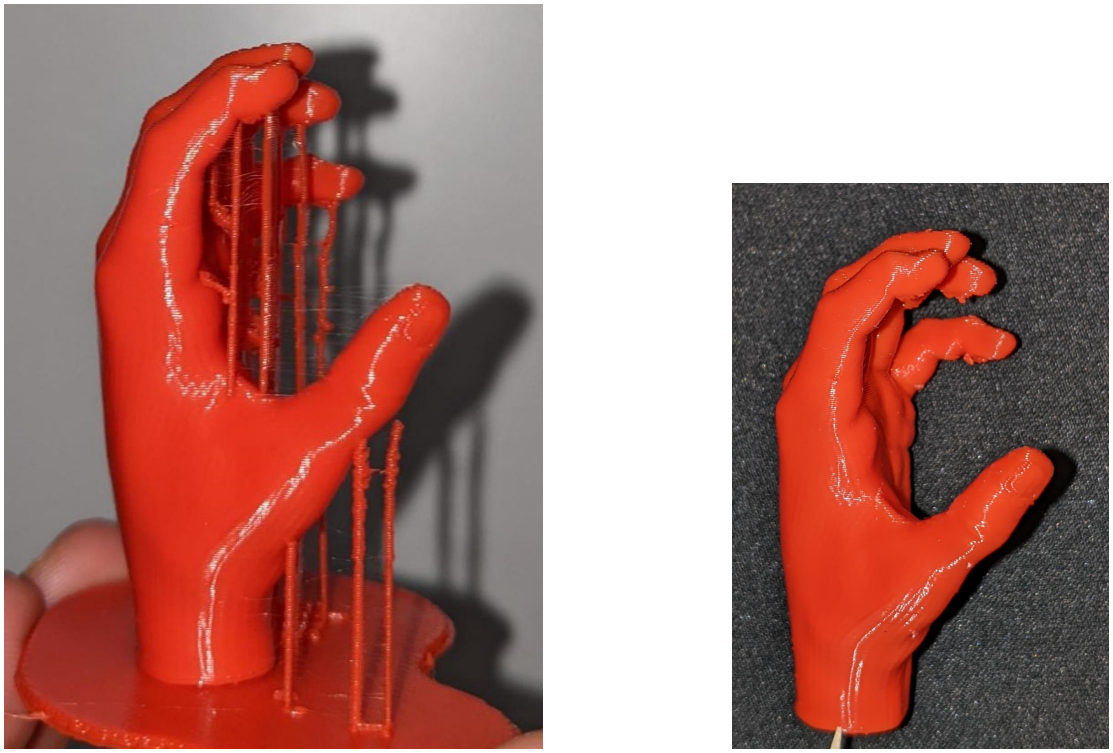

Figura 5.4: *Stampa: mano con e senza supporti.*

Figura 5.5: *Stampa: toro attorcigliato con e senza supporti.*

Appendice A

Reti Neurali

Una rete neurale (artificiale) è un insieme di neuroni (o nodi) suddivisi in strati (*layer*): un layer di input, uno di output ed uno o più nascosti (*hidden layers*). Le reti neurali supervisionate[1] vengono allenate fornendo in input dei dati e valutando tramite una *Loss Function* quanto l'output (ossia i parametri generati) sia lontano da quello desiderato, per poi applicare algoritmi di minimizzazione basati sulla discesa del gradiente per correggere i parametri. Questi due procedimenti (avanti e indietro) vengono chiamati rispettivamente *forward propagation* e *backpropagation*.
I parametri vengono elaborati tramite funzioni dette *funzioni di attivazione*, le quali agiscono su ogni singolo nodo a meno di quelli nell'input layer. Una delle prime strutture di reti neurali che ha portato risultati eccellenti è quella chiamata *Multi-Layer Perceptron* (MLP).

Multi-layer Perceptron

Struttura:

- Input layer: lo strato di input riceve i dati. Ogni neurone rappresenta una caratteristica del dato in ingresso;
- Hidden layer: uno o più strati intermedi tra input e output. I neuroni in questi strati ricevono input dai neuroni dello strato precedente e producono un output utilizzando una combinazione pesata e una funzione di attivazione. I pesi vengono inizializzati seguendo una distribuzione normale;
- Output layer: lo strato finale della rete produce un output che approssima la funzione cercata.

Funzionamento:

[1] Esistono anche non supervisionate, ma non vengono trattate in questa tesi.

- Fase di forward (Propagazione in Avanti): gli input vengono propagati attraverso gli strati nascosti fino allo strato di output e ogni neurone calcola una somma pesata dei suoi input, applica una funzione di attivazione f, e produce un output. Il processo continua fino alla generazione dell'output finale;
- Funzione di attivazione: ogni neurone, eccetto quelli dello strato di input, applica una funzione di attivazione non lineare $f(x)$. Alcune delle funzioni di attivazione più comuni sono:

$$\sigma(x) = \frac{1}{1+e^{-x}}, \quad \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}, \quad \mathrm{ReLU}(x) = \max(0, x);$$

- Calcolo della Loss Function: dopo che la rete ha generato un output, viene calcolata la loss function $L(y_{\text{pred}}, y_{\text{true}})$, che misura la differenza tra l'output previsto y_{pred} e quello reale y_{true};
- Backpropagation e aggiornamento dei pesi: viene ripercorsa la rete in senso opposto a quello precedente e tramite un algoritmo di discesa dei gradienti vengono aggiornati i pesi per ridurre l'errore determinato dalla loss function. I pesi iniziali vengono scelti seguendo una distribuzione normale perché altrimenti la discesa del gradiente non porterebbe alcun risultato.

Il MLP è una struttura di rete neurale che ha ottenuto risultati ottimi in tanti settori, tra cui: diagnosi mediche a partire da dati clinici, guida dei robot, problemi di classificazione e identificazione immagini. Per quanto riguarda, però, la generazione di immagini, il MLP con le funzioni di attivazione sopra citate, non riesce a produrre risultati apprezzabili.

Appendice B

Curve Spline

Definizione B.1 (Funzione Spline)**.** *Sia* $\Delta = \{x_i\}_{i=1}^{k+1}$ *una partizione dell'intervallo chiuso e limitato* $[a, b]$*, con*

$$a \equiv x_0 < x_1 < x_2 < \cdots < x_{k+1} \equiv b;$$

una funzione spline $s(x)$ *di grado* n *con nodi* x_i *per* $i = 1, \ldots, k$*, è una funzione su* $[a, b]$ *tale che:*

1. *in ogni sotto-intervallo* $[x_i, x_{i+1}[$ *con* $i = 0, \ldots, k$*, la funzione* $s(x)$ *è un polinomio di grado* n*;*
2. *la funzione* $s(x)$ *e le sue prime* $n - 1$ *derivate sono continue.*

Osservazione B.2. Una funzione spline è quindi una funzione polinomiale a tratti con la massima regolarità tra i nodi. Se questa non è necessaria, è possibile definire un vettore di continuità $M = \{\mu_i\}_{i=1}^{k}$ che indichi per ogni nodo quale sia il grado di continuità del raccordo.

Le funzioni spline posso essere definite come combinazione lineare di funzioni di base, dette B-spline, per le quali è necessario introdurre il concetto di partizione estesa.

Definizione B.3 (Partizione estesa)**.** *L'insieme* $\Delta^* = \{t_i\}_{i=1}^{2n+K+2}$ *con* $K = \sum_{i=1}^{k}(n - \mu_i)$ *si definisce partizione estesa se:*

- $t_1 < \cdots < t_{2n+K+2}$*;*
- $t_{n+1} = a; t_{n+K+2} = b.$

Definizione B.4 (Funzioni B-spline normalizzate)**.** *Sia* Δ^* *una partizione estesa dell'intervallo* $[a, b]$*. L'insieme delle funzioni B-Spline normalizzate* $\{N_{i,n}\}_{i=1}^{n+1+k}$ *è definito dalla formula ricorrente*

$$N_{i,n}(x) = \frac{x - t_i}{t_{i+n} - t_i} N_{i,n-1}(x) + \frac{t_{i+n+1} - x}{t_{i+n+1} - t_{i+1}} N_{i+1,n-1}(x),$$

dove

$$N_{i,0}(x) = \begin{cases} 1 & t_i \leq x < t_{i+1} \\ 0 & \textit{altrimenti} \end{cases}.$$

Si noti che la funzione di grado n può risultare in un quoziente $\frac{0}{0}$; in questo caso per convenzione viene associato il valore 0. È ora possibile definire una spline tramite le B-spline.

Definizione B.5 (Spline come somma di B-spline)**.** *Una funzione spline è una funzione che può essere rappresentata come combinazione lineare di funzioni base B-spline, ossia*

$$s(x) = \sum_{i=0}^{n} c_i N_{i,n}(x), x \in [a, b], c_i \in \mathbb{R}.$$

Definite le funzioni spline, possiamo estenderle a due (o tre) dimensioni, introducendo le curve spline.

Definizione B.6 (Curva spline 2D in forma parametrica)**.** *Dati i punti $P_i = (x_i, y_i)_{i=1,\dots,ncp}$, si definisce la curva spline di grado n e punti di controllo P_i:*

$$C(t) = \sum_{i=1}^{ncp} P_i N_{i,n}(t) = \begin{pmatrix} \sum_{i=1}^{ncp} x_i N_{i,n}(t) \\ \sum_{i=1}^{ncp} y_i N_{i,n}(t) \end{pmatrix}.$$

Dato il grado n e ncp il numero di punti di controllo, allora il numero di nodi interni è definito da $K = ncp - p - 1$ mentre il numero totale dei nodi della partizione estesa è $ncp + n + 1$.

Le curve spline presentano delle limitazioni in termini di rappresentazione; ad esempio non è possibile rappresentare le sezioni di coniche in maniera esatta. Per questo motivo vengono introdotte le curve NURBS (Non-Uniform Rational B-Spline).

Definizione B.7 (Curve NURBS)**.** *Le curve NURBS sono un'estensione delle curve spline e sono definite tramite la formula*

$$C(u) = \frac{\sum_{i=1}^{n_c} w_i P_i N_{i,n}(u)}{\sum_{i=1}^{n_c} w_i N_{i,n}(u)} = \left(\frac{\sum_{i=1}^{n_c} w_i x_i N_{i,n}(u)}{\sum_{i=1}^{n_c} w_i N_{i,n}(u)}, \frac{\sum_{i=1}^{n_c} w_i y_i N_{i,n}(u)}{\sum_{i=1}^{n_c} w_i N_{i,n}(u)} \right)^T.$$

Oltre agli elementi già visti per le spline, le NURBS presentano un vettore di pesi $(w_i)_{i=1,\dots,ncp}$ con $w_i > 0 \ \forall i$.

Osservazione B.8. Se $(w_i)_{i=1,\dots,ncp} = (1, \dots, 1)$ la NURBS diventa una spline non razionale.

Proprietà di una curva spline

Una curva spline di grado n in $[a, b]$ con nodi Δ e nodi aggiuntivi coincidenti gode delle seguenti proprietà:

1. La curva interpola il primo e l'ultimo punto di controllo:

$$C(a) = P_1, \quad C(b) = P_{n_{cp}};$$

2. La curva è continua e ha derivate continue di tutti gli ordini all'interno degli intervalli nodali ed è C^{h_i} nei nodi interni x_i;
3. La curva è tangente agli estremi al primo ed ultimo segmento del poligono di controllo.
4. La curva è contenuta nel *guscio convesso* formato dai suoi punti di controllo;
5. La curva è invariante rispetto a trasformazioni affini;
6. La curva ha la proprietà di *variation diminishing*: la curva spline non ha più intersezioni con una qualsiasi retta di quante ne abbia il suo poligono di controllo. Questa proprietà fa sì che le oscillazioni della curva siano al più quelle rappresentate dal suo poligono di controllo.

Questa trattazione non approfondirà ulteriormente le proprietà delle spline, ma basti sapere che quelle elencate garantiscono che le approssimazioni di superfici tramite spline sono regolari e convergenti. Tutte queste proprietà valgono ovviamente anche per le NURBS.

Bibliografia

[B] Botsch M., Kobbelt L., Pauly M., Alliez P., and Levy B., *Polygon Mesh Processing.*, A K Peters, Ltd, 2010.

[JB] Bloomenthal J., *Introduction to Implicit Surfaces.*, Kaufmann Publishers, 1997.

[C] Cybenko G., *Approximation by superpositions of a sigmoidal function.* Math. Control Signal Systems 2, 303–314 (1989). https://doi.org/10.1007/BF02551274

[JV] Jacquemet V., *Eikonal Equation: Computation*, In: Engquist, B. (eds) Encyclopedia of Applied and Computational Mathematics. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-70529-1_297

[N] NVIDIA, *GPU Gems 2 - Chapter 7. Adaptive Tessellation of Subdivision Surfaces with Displacement Mapping*, https://developer.nvidia.com/gpugems/gpugems2.

[S94] Samet H., *The Design and Analysis of Spacial Data Structures.* Reading, MA: Addison Wesley, 1994, p. 94.

[S] Sitzmann V., Martel N. P. J., Bergman A.W. , Lindell D. B., Wetzstein G., *Implicit Neural Representations with Periodic Activation Functions*, NeurIPS 2020. https://arxiv.org/abs/2006.09661

[S3D] *The Stanford 3D scanning repository*, last modified 6 Apr 2023 23:57:40 CDT, https://graphics.stanford.edu/data/3Dscanrep/.

Ringraziamenti

Ringrazio i miei genitori che mi hanno permesso di studiare in totale libertà, finanziando e supportando la mia carriera universitaria in ogni momento.
Grazie a Susanna, che mi ha aiutato per tutta la triennale ascoltando interessata spiegazioni di argomenti che risulterebbero incomprensibili anche dopo mesi di studio, e che non si è mai tirata indietro quando avevo bisogno di aiuto.
Grazie ai miei coinquilini, Bata e Gube, che hanno sopportato il rumore dei tasti del pianoforte anche a mezzanotte e, comunque, non si sono mai lamentati.
Tutti gli altri miei amici, un elenco sarebbe troppo lungo, sanno che confrontarmi con loro mi ha permesso di mantenere alta la motivazione e di non perdere mai di vista l'obiettivo.
Per ultimi, ma solo perché con qualcuno la lista sarebbe dovuta finire, ringrazio i compagni del Falansterio, senza i quali la laurea a settembre sarebbe rimasta semplicemente un sogno.