T18- from 2D to 3D Modelling

Trova applicazioni nella computer graphics, nella modellazione 3D e nella robotica.

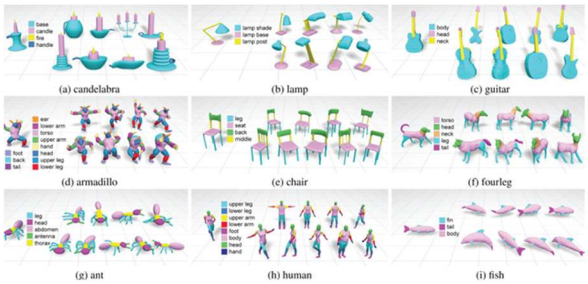
Applicazioni ed esempi:

- Mi permettono di vedere la relazione che sussiste tra il tipo di dato in ingresso nella rete neurale e l'architettura della rete stessa -> quali architetture sono più adatte ad un tipo di dato
    - Quali sono le proprietà che fanno in modo che una rete apprenda più facilmente -> converga più velocemente ad un dato accettabile?

Cosa portarsi via da oggi:

- Quali sono le rappresentaizoni 3D?
- Quali sono i loro vantaggi?
- Quali sono i loro svantaggi?
- Quali sono le implicazioni che hanno sulla rete?

Alcuni esempi:

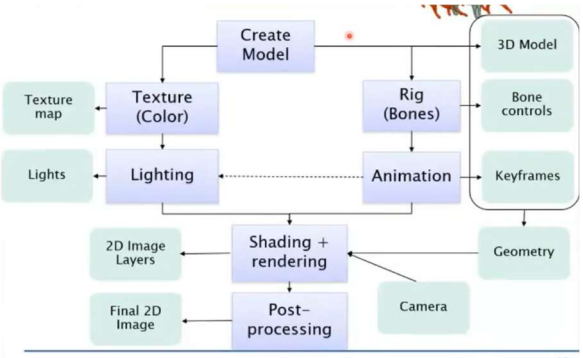


- **Mesh labelling**: se abbiamo una mesh (modello di una superficie tridimensionale), potremmo pensare di etichettare le varie parti della mesh in relazione alla parte dell'oggetto -> segmentazione
- **Pixel2Mesh**: voglio partire da un dato bidimensionale ad un oggetto tridimensionale -> la rete dovrà fare un'ipotesi su come completare l'oggetto basandosi sulle statistiche di oggetti simile che conosce

- **Rendering:**
- **Animation:** generazione da parte di una rete neurale a partire da una traccia audio.
- **Style transfer** applicato a dati tridimensionali con una componente temporale -> quadridimensionale
- **Automotive:** processamento di point Cloud che vengono generati da dei sensori
    - **Sintetici:** da modellazione 3D
    - **Da acquisizione:** provenienti da sensori
- **Addictive manudacturing:** dati ti tipo volumetrico

Deep Learning in Computer Graphics

Mondo dell'animazione funziona così:



- Modellazione tridimensionale:
  - Superficie a cui viene applicata una texture che rappresenta il colore e il tipo di superficie, con delle fonti di luce per creare un'immagine simile a quella che l'occhio umano percerirebbe nella vita reale.
  - Questo avviene tramite un'operazione di rendering + shading -> ottengo immagini bidimensionali tenendo conto delle caratteristiche instrinseche della superficie e del modo in cui interagisce con la luce. (riflettenza + irradazione)
  - A questo può seguire una fase di post-processing per migliorare la qualità dell'immagine renderizzata e ridurre la rumorosità dell'immagine.
  - Il modello è caratterizzato da:
    - Rig: ossatura del modello -> coni che possono essere mossi per rappresentare posizione -> uso per fare animazione.

Nel machine learning, tipicamente si ragiona per task: apprendimento di una funzione che mappa uno spazio di input X e uno spazio di output Y.

- Voglio capire quali sono gli spazi di input e quali quelli di output per poter capire quale tipo di reti neurali devo usare.

Problem	Input → Output	Input → Output
Shape classification	3D model → Label	$\mathbb{R}^{3d} \rightarrow \mathbb{R}^{k}$
Denoising, smoothing	Image → Image	$\mathbb{R}^{m \times m} \rightarrow \mathbb{R}^{m \times m}$
Inverse graphics	Photo → 3D model Photo → Rendering	$\mathbb{R}^{m \times m} \rightarrow \mathbb{R}^{3d}$ $\mathbb{R}^{m \times m} \rightarrow \mathbb{R}^{m \times m}$
Animation	3D model + time → 3D model	$\mathbb{R}^{3d \times t} \rightarrow \mathbb{R}^{3d}$
Generative models	Latent space → 3D model	$\mathbb{R}^{k} \rightarrow \mathbb{R}^{3d}$

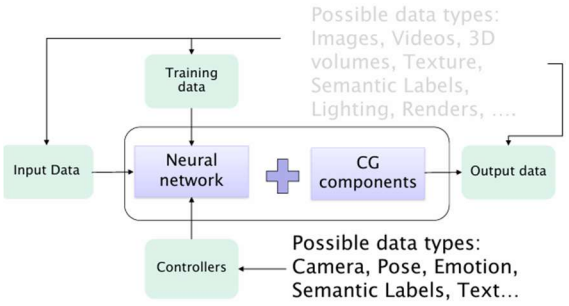
->È una bottiglia o un portafoglio?

->Migliorare un immagine

-> da immagine a 3D

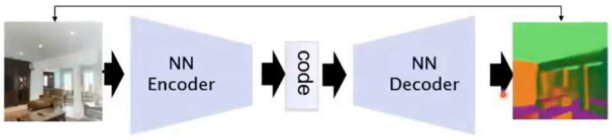
Nella computer graphics, posso includere nella stessa pipeline dei componenti neurali con componenti non neurali:

- **Neural rendering**: implica la possibilità di sostituire una qualunque (o + fasi) con una rete neurale (sostituisce tutta o una parte del rendering)
  - Maggior realismo -> reti neurali mi permettono di apprendere dai dati alcune statistiche
  - Ridurre il tempo -> alcune fasi possono essere automatizzate
  - Solitamente, in molte pipeline, esiste un componente CG che interviene tra l'output della rete neurale e il dato finale:
    - Può essere differenziabile o non differenziabile
      - Se differenziabile, posso retropropagare i gradienti e quindi posso aggiungere la componente di computer graphics anche nel training -> come d
      - Se fosse parte della funzione di loss del nsotro modello



Vari casi:

- **I e O: mappe bidimensionali** -> tipologia di problema che si può ricondurre ad una segmentazione semantica -> può essere risolta con una rete di tipo encoder-decoder.
    - Riadatto una delle reti viste per semantic segmentation (cambiando opportunamente la loss)
    - La funzione che viene appresa dipende da come generiamo il segnale di supervisione:
        - Come generiamo e acqisisco il segnale
    - Nell'ambito della CG ci sono delle applicazioni dove si arricchisce il modello usando principi fisici legati al modo in cui sappiamo si forma l'immagine (code)



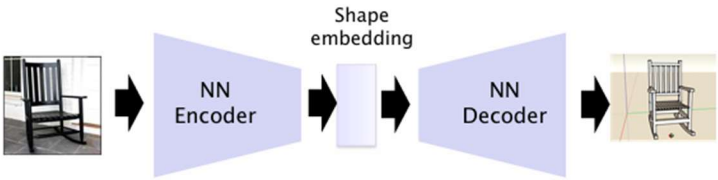
- Esempio denoising sulle slide dove l'informazione di dominio viene aggiunta -> modello si specializza.
- **Grafica inversa**: separare un'immagine nelle sue componenti fondamentali:
    - **Riflettanza:** Il colore intrinseco delle superfici, indipendente dall'illuminazione.
    - **Illuminazione (Shading)**: Le variazioni di luminosità dovute alle condizioni di luce e alla geometria della scena.

  Applicazioni pratiche:

    - Cambiare colore oggetto
    - Cambiare fonte di illuminazione

  In una foto arbitratia, non è presente una rappresentazione standard delle due componenti, dobbiamo trovare un riferimento standard :

    - Esempio annotazione degli umani -> differenziato pixel chiaro vs scuro.
    - Utilizzare dei dati sintetici -> Un'architettura per la decomposizione intrinseca delle immagini che produce caratteristiche **normali** (geometriche della scena) e **albedo** (colore intrinseco della superficie), e le ricombina attraverso un **modulo neurale leggero per stimare la luce** (light estimator)
        - Partendo dalle componenti voglio arrivare all'immagine: uso dati sintetici per generare il mio riferimento standard.
        - Sono intrinsicamente annotati ma non sono realistici.
        - Ad una prima generazione su immagini sintetiche, bisogna fare seguire un fine-tuning su immagini reali (o su etichette effettivamente acquisite se mi trovo in un problema di classificazione semantica) oppure fine-tuning auto-supervisionato con la rete neurale che genera le proprie etichette e si fa un passo di supervisione sulle etichette generate.
- **Da immagine a modello 3D**
  Si deve partire da più punti di vista a causa della auto-occlusioni.



Rappresentazioni 3D

- Euclidee: rappresentazione organizzata su uno spazio regolare
    - Immagine bidimensionale -> griglia di pixel con indici per coordinate
    - Immagini RGB-D: uso CNN o transformer
    - Multiview: da 3D -> 2D usando CNN o transformer
    - Volumetrico: esensione tridimensionale dell'immagine bidimensionale -> la nostra griglia diventa un tensore di m*m*n : uso CNN per dati 3D
    - Descrittori: si basano sul modello parametrico -> una volta ottenuti i descrittori, si ha in ingresso un modello di feature.
- Non euclidee: punti non riconducibili ad una rappresentazione regolare nello spazio
    - Nuvole di punti
    - Grafi: rappresentato da una serie di nodi collegati da archi a cui sono associate delle feature
    - Meshes: combinazioni di forme regolari come triangoli

Modelli Parametrici

I modelli parametrici offrono una rappresentazione semplificata degli oggetti 3D, con l'obiettivo di descrivere caratteristiche geometriche o topologiche fondamentali. L'applicazione più diffusa è quella della modelizzazione del corpo umano -> SMPL.

SMPL (Skinned Multi-Person Linear Model)

Descrizione

SMPL è un modello parametrico avanzato progettato per rappresentare forme umane. Combina forti **priori anatomici** con una rappresentazione matematica compatta.

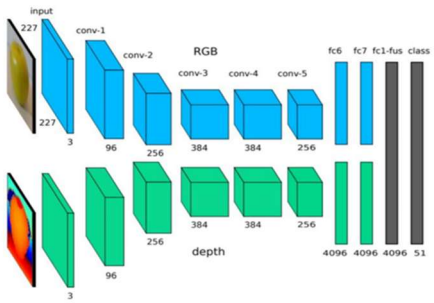
Caratteristiche principali

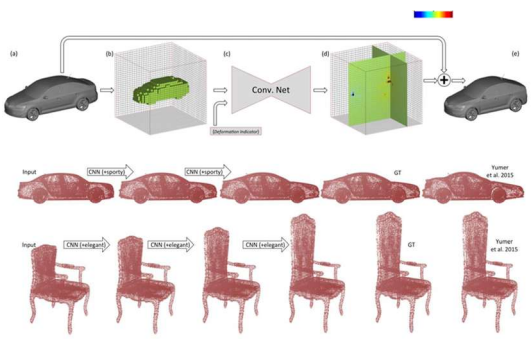
1. **Parametri di controllo**:
    - **72 parametri** controllano pose e forma:
        - **Pose**: 3 parametri per ogni articolazione (rotazioni 3D).
        - **Shape**: descrizione lineare delle variazioni del corpo.
2. **Completamente differenziabile**:
    - Adatto per l'apprendimento end-to-end, utilizzando gradienti per aggiornare i parametri.
3. **Applicazioni**:
    - **Stima del movimento umano**: ricostruzione accurata delle pose a partire da immagini o video.
    - **Motion capture**: integrazione nei sistemi di cattura del movimento.
    - **Animazione**: generazione di movimenti realistici per personaggi virtuali.

Multiview – RGBD:

Acquisiscono immagini da più punti di vista.

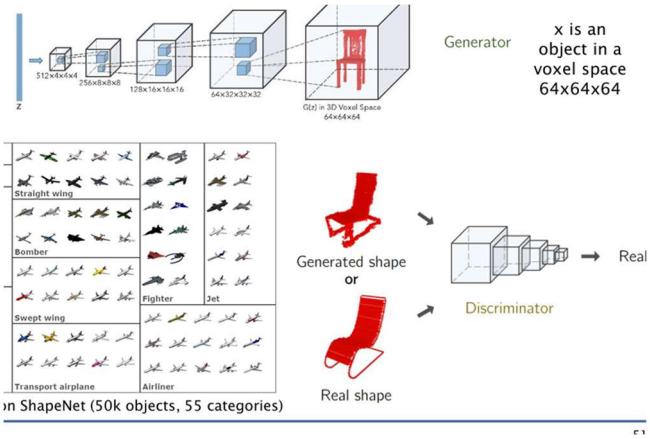
- Esempio RGBD: convnet + fusione dei due stream per produrre classificazione finale





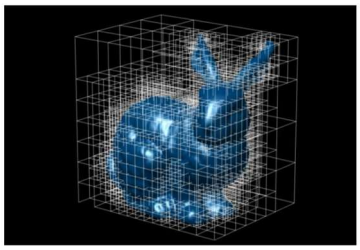
- Dati volumetrici: posso usare del convnet 3D: rete deve predirre se il corrispondente box è pieno o vuoto

➔ Può essere usato da un modello generativo per generare delle forme.



Nota: il problemà principale è la densità -> volumi troppo sparsi, la pecentuale di pixel pieni è bassa.

- Si è provato ad usare octree -> box più grossi (vuoto vs pieno)
    - E poi gerarchicamente -> non posso più usare le 3D convnet



- Image-based (Multi-view, RGB-D)
    - Pros: directly use image networks, good performance
    - Cons: rendering is slow and memory-heavy, not very geometric
- Volumetric (Voxels, Octree)
    - Pros: 3D CNN are direct extension of 2D CNNs
    - Cons: very large and sparse data structures (volumes), special layers for hierarchical data structures (octrees)

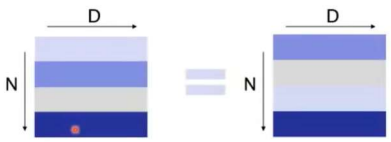
DATI NON EUCLIDEI

- **Point Cloud**: rappresentazioni 3D (nota: non posso usare le conv3D)
  Abbiamo varie possibilità:
  - Ricondurci ad un'altra rappresentazione nota
    - Volume euclideo: associo le feature della point cloud al punto corrispondente dalla posizione, lasciando vuoti tutti i punti che non sono occupati dalla point cloud
  - Varianti specifiche
    - Rete PointNet: pensata esplicitamente per lavorare direttamente su point cloud
- **Grafi**
  - Ci sono reti neurali che sono state appositamente pensate per i grafi che reinterpretano l'operazione di convoluzione per lavorare sui grafi -> non le vediamo
- **Mesh**
  - Possono esssere elaborate con una variante della CNN adatta per lavorare sulle mesh o possono essere convertite nelle rispettive PointCloud/Grafi usando i vertici dei triangoli come nodi del grafo / punti della point cloud (considerando le mesh triangolari).
    L'operazione inversa è più complessa invece.

POINT CLOUD

Rappresentazione molto comune per i dati 3D: sia per dati sintetici che per dati reali.

Sono rapprewsentazioni non strutturate, rappresentate da un elenco di N punti



- Ciascun punto è associato a D coordinate
- Più **compatta** della rappresentazione volumetrica perché mi consente di salvare soltanto quella piccola percentuale di pixel pieni
- Rappresentazione **non strutturata**: qualsiasi permutazione mi rappresenta la stessa nuvola di punti -> quindi qualunque rete neurale io voglia usare, deve essere consapevole che questi punti non hanno una sequenza intrinseca.
  - RNN può lavorare su una sequenza di punti di questo tipo ma questa è stata pensata per lavorare su sequenza (processa une lenmento per volta e assume che ci sia un ordinamento all'interno della sequenza) ma nella point cloud non c'è un ordinamento.
  - TRANSFORMER possono lavorare su questo tipo di sequenze in quanto non fanno alcuna assunzione sull'ordine (infatti per gestire le immagini abbiamo dovuto usare i positional embedding).
  - POINTNET: Rete neurale speicifica per questo tipo di dati, presenta (codificate all'interno della rete) delle invarianze che sono utili per trattare questo tipo di dati.

Caratteristiche della point cloud:

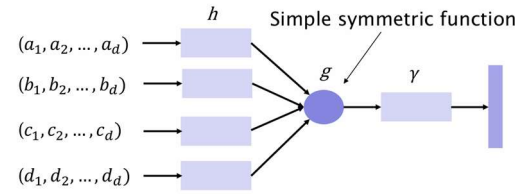
- **Invarianza rispetto alla permutazione**:
  - Se abbiamo N punti, questi N punti possono essere permutati in N! permutazioni, tutte queste permutazioni rappresentano la stessa forma e l'output della rete deve essere inviariante rispetto al particolare ordine in cui la point cloud viene processata.
    Se non la codifico nella rete, l'invarianza deve essere appresa dai dati con metodi di data-augmentation facendo vedere alla rete la stessa point-cloud con diverse permutazioni in modo tale da insegnare alla rete che quelle permutazioni sono la stessa point cloud.
- **Invarianza rispetto alle trasformazioni geometriche**:
  - Le rappresentazioni che imparo devono essere invarianti rispetto alle trasformazioni rigide: rotazione e traslazione. Dunque, se io sposto un oggetto nello spazio o lo ruoto, le sue caratteristiche rimangono invariate.
  - Nota: le CNN sono equivarianti rispetto alla traslazione ma non rispetto alla rotazione a causa dell'immagine che è 2D. In un oggetto tridimensionale vorrei che ci sia univarianza rispetto alla rotazione.
- **Iterazione tra i punti locali**:
  - All'interna di una nuvola di punti, ci sono delle nuvole che scaturiscono dalle interazioni dei punti vicini. Si viene a formare una struttura gerarchica per cui i punti vicini formeranno delle parti che poi si legano per formare un oggetto.

Come rendere la rete invariante rispetto all'ordine delle point-cloud?

- Devo usare una funzione matematica che soffisfi queste proprietà.
- Tutte le funzioni simmetriche mi permettono di ottenere invarianza rispetto alla permutazione
    - Funzioni in cui il risultato, non dipende dall'ordine degli elementi che vengono presi in ingresso dalla funzione stessa. (es: somma)

Osservazione su **composizione di più funzioni**:

Observation:
$f(x_1, x_2, \dots, x_n) = \gamma \circ g(h(x_1), h(x_2), \dots, h(x_n))$
is symmetric if g is simmetric



h: applicata in maniera indipendente a tutti gli elementi, prende in ingresso ciascun elemento della point cloud.

g: combina in maniera opportuna i vari output della funziona h e viene data in ingresso a ɣ

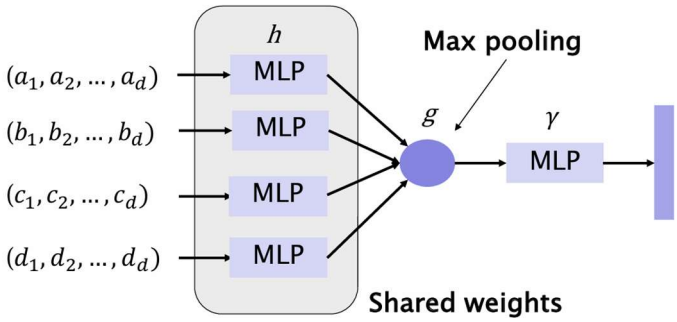
ɣ: computa l'output finale -> rete neurale che prende in ingresso dei vettori -> multi-layer-perception

In questo caso, la funzione *f* è simmetrica se *g* è simmetrica.

Come costruisco la funzione g?

- Possiamo usare una funzione di somma in quanto simmetrica
- Massimo
- Media
- Prodotto -> genera problemi legati ai gradienti

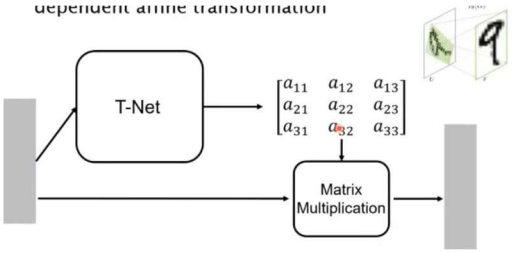
Nella Vanilla PointNet si usa il massimo
Empirically, simple functions such as **multi-layer perceptron (MLP)** and **max pooling** are effective

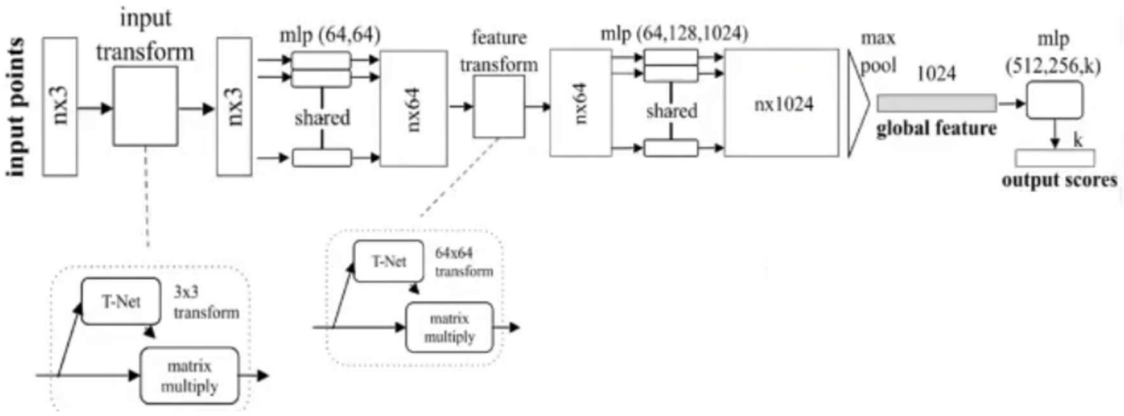


Per h e ɣ, si usano solitamente delle MLP: reti a più strati con layer FC, intervallati con funzioni lineari, di solito con shared wiight, quindi ciascun punto viene processato dalla stessa funzione.

Invariante rispetto alla permutazione in quanto la funzione h che processa ciascun punto è la stessa.

Non è invariante, di per se, rispetto alla rotazione, dunque nella pointnet viene introdotto un altro componente con il compito di imparare una trasformazione (tipicamente roto-traslazione) che viene applicata all'ingresso della rete in modo da allineare tutte le point-cloud rispetto allo spazio di riferimento -> Special Transormer

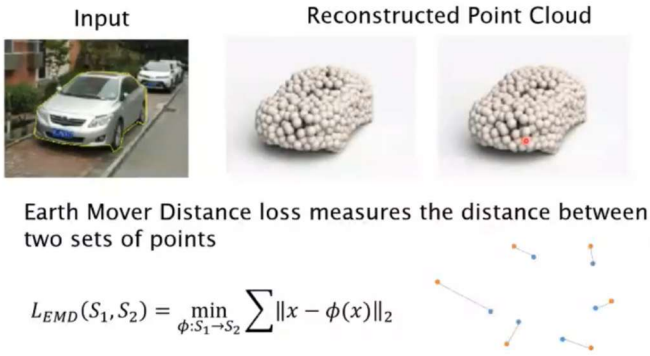




Nota: Pointnet può essere usata per problemi di segmentazione, modificando l'ultima parte della rete -> invece della MLP, uso una rete più complessa che mi da un output per ogni pixel.

In questo tipo di rete, lo spazio di output sono label -> classificazione/Segmentazione (posso usare come loss cross-entropy)

Nel caso in cui lo spazio di output sia una point-cloud -> rete generativa che sintetizza le point cloud o rete che fa un psot-processing/trasformazione di una point cloud o rete che partendo da immagine 2D generi una point cloud (3D).



- Voglio misurare la distanza tra point-cloud generata e quella di riferimento
- Non ci troviamo in uno spazio euclideo quindi non abbiamo un riferimento ovvio
    - Abbiamo 2 point-cloud, ciascuna composta da N elemento ma non è detto che il primo elmeento della prima point-cloud corrisponde al primo elemento della seconda in quanto non esiste un ordine e non esiste un sistema di coordinate -> dobbiamo trovare una misura di distanza che sia applicabile e che ricostruisca in maniera automatica la miglior accoppiata tra i punti della point cloud generata e quella di riferimento, diversi tipi di LOSS:
        - **Lemd**: quanto materiale dovrei spostare per ottenere la point cloud di riferimento partendo da quella generata (Accoppio punto di quella generata con quello a lui più vicino dell'altra)

Questi tipi di point cloud visti finora derivavano da scansioni di oggetti visti a 360°. Esistono anche altri tipi di point cloud, tipicamente acquisiti in ambiente outdor che rappresentano una scena e che sono molto più grandi e sparse. L'esempio tipico di questo tipo di point-cloud sono quelle acquisite da sensori di tipo **LIDAR**: sensori tipicamente usati nella guida autonoma e che vanno a scansionare lo spazio circostante ad un veicolo. PointNEt potrebbe non essere così efficiente per questo tipo di dati a causa della dimensione delle point cloud; inoltre tipicamente bisogna unire queste informazioni con quelle di altri sensori.

- Una possibilità è quella di proiettare la point-cloud in una sfera e poi proiettare la sfera su un piano.
    - Ad ogni punto dell'immagine, associo intensità e distanza misurata e sull'immagine bidimensionale effettuo l'operazione che devo fare (mi sono ricondotto ad un'imamgine 2D)
- Altra possibilità è generare una Bird-Eye view in cui vado a vedere la point cloud dall'alto, proietto punti visti dalll'alto su delle mappe bidimensionali -> questa visione facilita la fusione con delle mappe

Come processare ddei dati volumetrici nel tempo:

- **Animazioni**: serie temporale di pose

Una posa può essere rappresentata come:

- Modello parametrico: abbiamo delle feature continue che rappresentano i parametri del modello
    - Continous
    - Discrete
- Joint position: se abbiamo delle sequenze di punti, le cui posizioni possono esssere assolute o relative e possono essere rappresentare come:
    - una sequenza di numeri (feature)
    - un grafo dove i vari giunti sono rappresentati d aun nodo nel grafo e gli archi rappresentano le connessioni anatomiche o funzionali tra i vari giunti
        - ci serve una rete neurale che sia in grado di elaborare un grafo
        - bisogna far attenzione a non inserire troppi vincoli in un sistema:
            - rappresentazione atomica potrebbe non essere la più efficace, solitamente si aggiungono anche alcune connessioni funzionali tra giunti che non sarebbero di per se direttamente collegati
    - alcuni task possono essere:
        - motion prediction/syntesis
        - interaction prediction/syntesis

Tecniche di apprendimento che possono essere utilizzate, per predirre una sequenza di parametri basati su posizione di giunti:

- se abbiamo un problema supervisionato:
    - usare una loss di regressione.
        - Reti ricorrenti
        - Reti convoluzionali
        - Reti basate su grafo

Ci sono altre tecniche di apprendimento basate su Reinforcement learning