

Point Cloud

Grazie alla presenza di normali, che descrivono l'orientamento della superficie, possiamo immaginare la nuvola di punti come una collezione di «surfels» (pezzetti orientati di superficie) piuttosto che di punti (non orientati)

VEDERE LA LEZIONE
 Su algebra di punti e vettori

14

Storage di point cloud

✓ Esempio di formato di file per point cloud: <filename>.xyz

Posizione del primo campione (x y z)

Normale del primo campione (x y z)

```

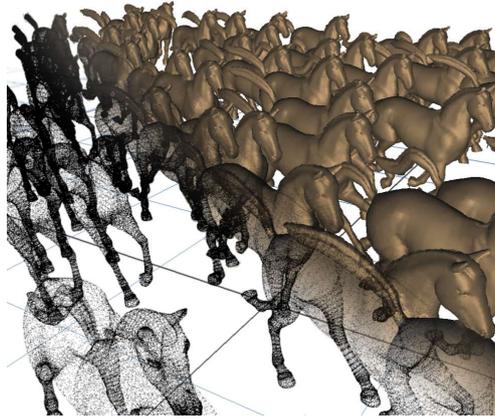
-93.588310 384.453247 8.672122 -0.750216 0.605616 0.265339
-93.365311 384.456879 9.228838 -0.766408 0.558148 0.317947
-92.981407 384.595978 9.903000 -0.771328 0.544261 0.329898
-93.032166 384.662994 9.664095 -0.753935 0.579665 0.309145
-92.670418 384.707367 10.423333 -0.773636 0.510460 0.375390
-92.432343 384.866455 10.697584 -0.775882 0.504696 0.378535
-92.233391 384.935974 11.023732 -0.770233 0.538780 0.341258
-91.959816 384.952209 11.629505 -0.765533 0.534614 0.357975
-91.684883 385.122528 11.965501 -0.783344 0.499727 0.369656
-90.981750 385.299408 13.283755 -0.797000 0.474158 0.374119
-91.017151 385.362061 13.132144 -0.801652 0.457642 0.384600
-90.481400 385.523560 13.933455 -0.691992 0.589880 0.416160
-90.241745 385.597351 14.216219 -0.664425 0.625032 0.409725
-89.772568 385.683197 14.820392 -0.626528 0.653291 0.425056
-89.167023 385.833191 15.455617 -0.654831 0.640269 0.401562
-88.530830 386.009369 16.353712 -0.737686 0.498695 0.455108
-87.759621 386.192017 17.259428 -0.681486 0.572231 0.456211
-86.996521 386.357880 18.085392 -0.653592 0.574349 0.492890
-86.823006 386.214783 18.469635 -0.665109 0.533859 0.522135
-85.953079 386.573792 19.228466 -0.682286 0.524974 0.508810
    
```

...eccetera

16

Point cloud: rendering

- ✓ Attraverso «point splats»:
- ✓ «splat» = una regione di pixel sullo schermo che vengono attivati per rappresentare il punto



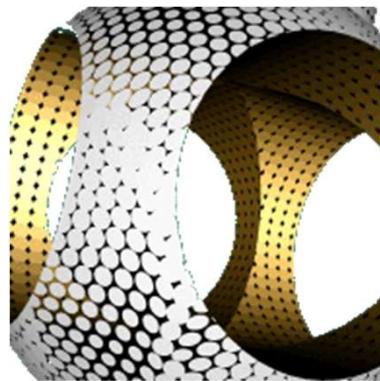
img by Michael Wimmer 2012



17

Point cloud: rendering

- ✓ la dimensione dello splat viene scelta in funzione della densità dei punti, la distanza media fra loro
- ✓ disegnabili anche come dischetti (pezzetti di superficie)



img by Mario Botsch, 2004



18

Multirisoluzione su point cloud

- ✓ La point-cloud è semplice da gestire perché ogni punto è indipendente dagli altri
- ✓ Ad esempio, una semplice **multirisoluzione** si ottiene semplicemente prendendo un sottoinsieme arbitrario dei punti
- ✓ **Strategia:** (adottata ad es. da «qSplat» - Stanford uni)
 - ⇒preordinare la point cloud di N vertici in modo che i primi $M < N$ vertici siano ben distribuiti, per ogni M
 - ⇒(un random shuffle è buona approssimazione!)
 - ⇒in fase di rendering, disegnare solo i primi M vertici per avere una point cloud di risoluzione M

19

Multirisoluzione su point cloud



131,712 splats 132 ms



259,975 splats 215 ms



1,017,149 splats 722 ms



14,835,967 splats 8308 ms

Images by QSplat software

22

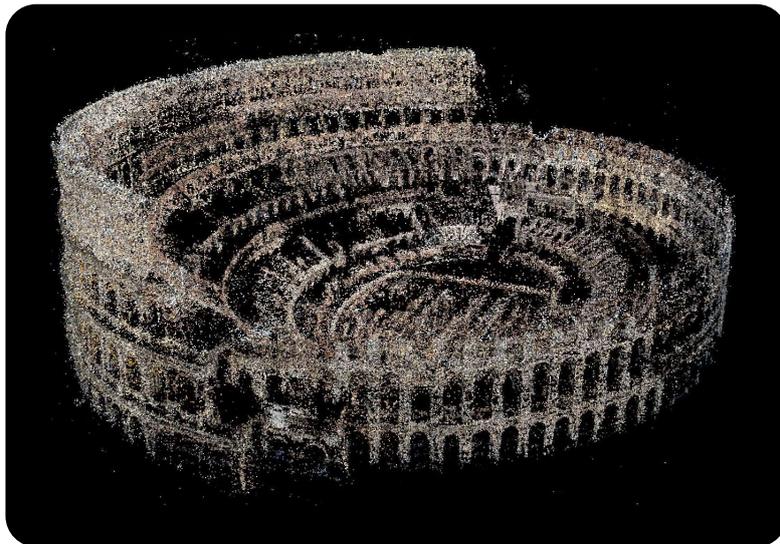
Geometry Processing on point clouds

- ✓ Esistono tecniche di creazione di point cloud a partire da immagini
 - ⇒ Shape From Motion / Photogrammetry (problemi studiani in seno alla Computer Vision)

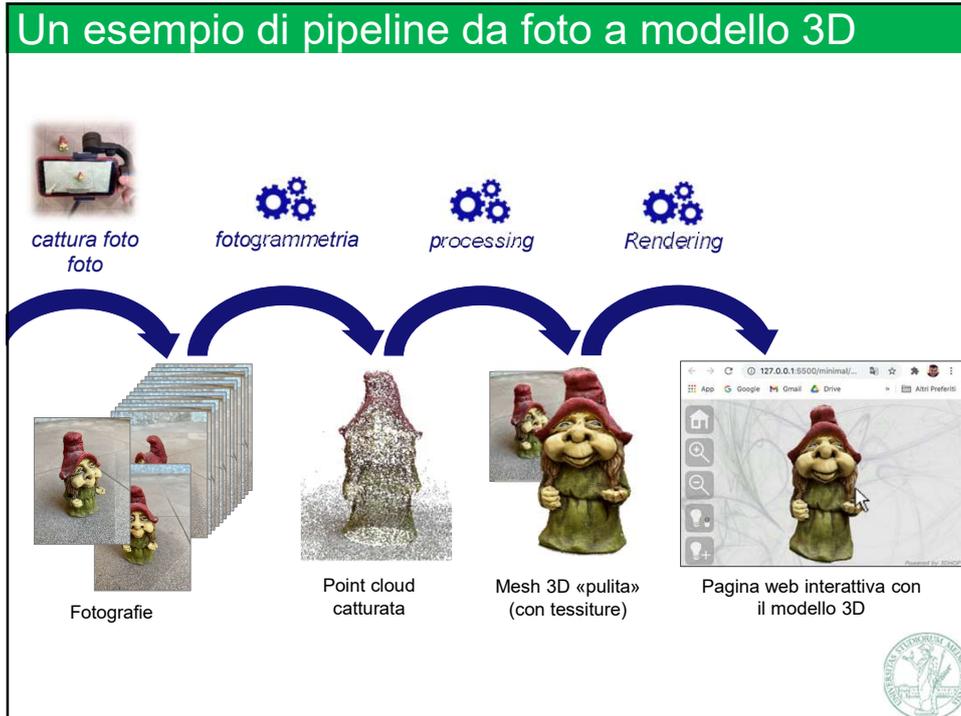


23

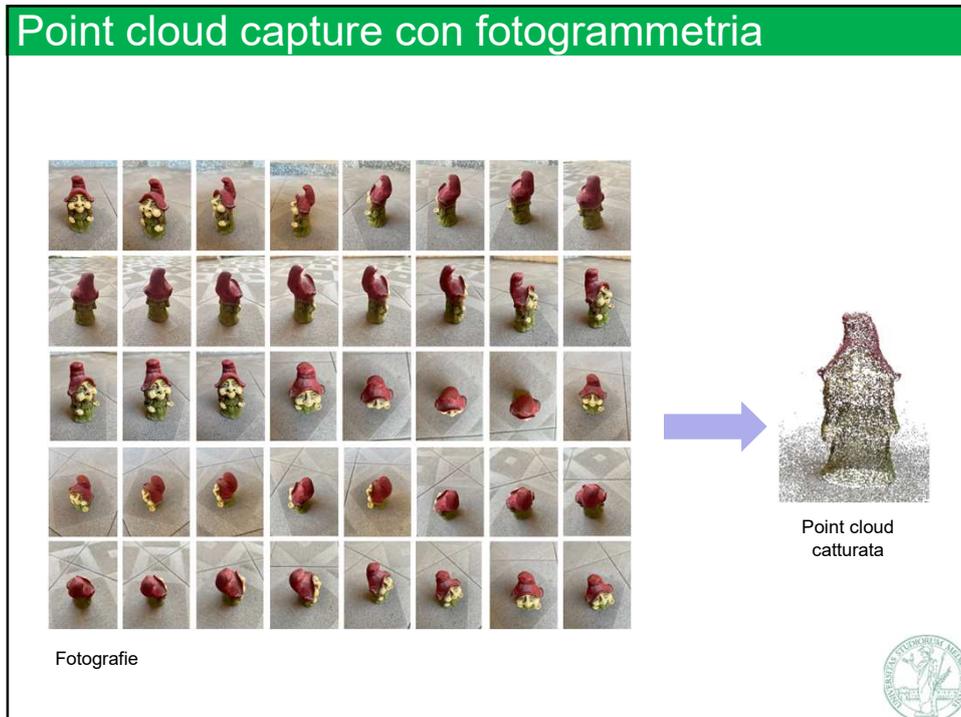
Una point cloud acquisita da fotografie



24



25



26

Alcuni software e strumenti esistenti

✓ Per cattura di fotogrammetria

✓ Un repository di point-clouds catturate:



<https://sketchfab.com/tags/point-cloud>



27

Alcuni software e strumenti esistenti

Per geometry-processing effettuato su point clouds



Point Cloud C++ Library

MeshLab

Cloud Compare

specializzato anche in range scans, un modello di dato che vedremo più tardi

vedi:
«Filters» → «Point Set»



28

Geometry Processing su point clouds: k-NN

- ✓ Un sotto-problema ricorrente:
trovare il **vicinato spaziale** di ogni vertice
 - ⇒ Cioè rispondere alla domanda:
dato un vertice, quali sono i vertici a lui più vicini?
 - ⇒ Molto del processing su point-cloud richiede di risolvere questo come sotto-problema
- ✓ Definizione del problema:
 - ⇒ «dato un vertice i in posizione \mathbf{p}_i ,
trovare tutti i vertici j che siano in una posizione \mathbf{p}_j
entro una certa distanza d_{MAX} da \mathbf{p}_i »
 - ⇒ dove d_{MAX} è un parametro del problema
 - ⇒ (sapresti esprimere questa condizione con un'espressione dell'algebra di punti e vettori?)
- ✓ Questo modo di definire un vicinato ha molti problemi
 - ⇒ il valore d_{MAX} dipende dalla scala del modello (è *scale dependent*)
e dalla sua risoluzione
 - ⇒ nessun parametro (costante) funziona bene con risoluzione adattiva



30

Geometry Processing su point clouds: k-NN

- ✓ Una *migliore* definizione del problema:
 - ⇒ dato un vertice i in posizione \mathbf{p}_i ,
trovare i k vertici più vicini a lui
 - ⇒ dove k è un parametro del problema: per es, 10, o 8
 - ⇒ questi k punti sono detti i **k nearest neighbors**
(in sigla: **k-NN**)
 - ⇒ Perché questo modo di definire un vicinato è molto più robusto?
(ed è quello adottato praticamente in ogni situazione)
- ✓ Problema: efficienza
 - ⇒ In ogni caso, risolvere questo task con tecniche «forza bruta»
non è efficiente
 - ⇒ «Forza bruta» (in questo caso) = testare tutti i campioni
 - ⇒ Trovare il vicinato di un punto: complessità lineare (con la risoluzione)
Trovare il vicinato di tutti i punti: complessità quadratica
 - ⇒ (vedi corso di algoritmi per la spiegazione di questi termini!)
 - ⇒ INFO: in CG, una complessità quadratica, è considerata proibitiva
 - ⇒ Sono necessarie tecniche algoritmiche più sofisticate (che non vedremo)



31

Geometry Processing su point clouds: stima delle normali

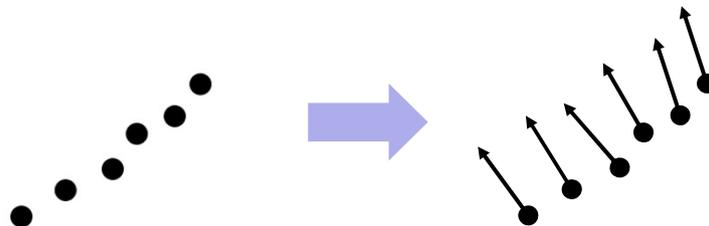
- ✓ Una point cloud può essere sprovvista di direzioni normali per ogni campione
- ✓ E' dunque necessario stimare le normali a partire dalle posizioni
 - ⇒ Vedremo, questo è un problema ricorrente anche con altri tipi di dato
- ✓ Come? (schema di una soluzione)
 - ⇒ Per ogni punto:
 1. trovare il suo vicinato (k-NN)
 2. trovare il piano passante per il suo vicinato («best fitting plane»)
 3. assegnare a quel punto la normale di quel piano
 - ⇒ Nota: un piano passa per tre punti, ma nessun piano in generale passa per $N > 3$ punti maggiore. Tuttavia, è possibile trovare il piano che *minimizza la distanza da N punti dati* (problema risolvibile usando l'algebra di punti e vettori)
 - ⇒ Nota: è necessario risolvere anche un'ambiguità di direzione: un piano ha due normali di segno opposto



33

Geometry Processing su point clouds: stima delle normali

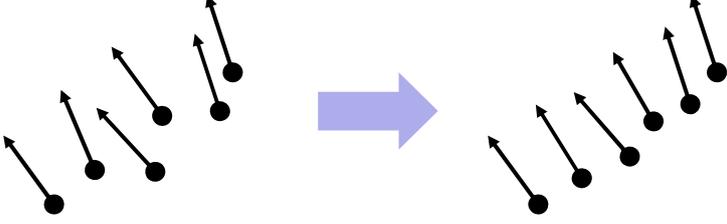
- ✓ Stima delle normali su una point cloud



34

Geometry Processing su point clouds: denoising

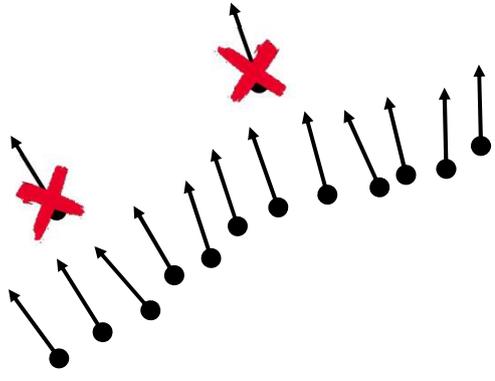
- ✓ Denoise – Noise reduction – o «fairing»
⇒ Di posizioni (e/o normali)



35

Geometry Processing su point clouds: outlier removal

- ✓ Rimozione degli «outlier»



36

Geometry Processing su point clouds: denoising

- ✓ Come tipico per altri modelli 3D acquisiti dalla realtà, errori di misurazione si traducono in «rumore» sul dato
 - ⇒ nel nostro caso: si tratta di errori nella posizione e nella normale
 - ⇒ è anche il caso, ovviamente, per dati catturati dal vero di ogni altro tipo come immagine o dati auditivi (da cui il nome di «rumore» -- noise)
- ✓ Un task tipico è dunque de-noising: rimuovere il «rumore» dalla point cloud, tentando di preservare il «segnale»
 - ⇒ Spostare ogni campione in modo da avvicinarlo alla superficie reale
- ✓ Concetto generale per point-cloud de-noising:
 - ⇒ Si suppone che le superfici tendano ad essere lisce: superfici molto frastagliate devono essere rese più lisce avvicinando ogni punto al piano passante per il suo vicinato
- ✓ Un problema solo simile è quello della rimozione degli outliers:
 - ⇒ Alcuni punti possono essere stati del tutto «inventati» dal processo di ricostruzione 3D (per errori, o perché sono realtivi ad altre superfici): questi punti non devono essere rimossi, non modificati
 - ⇒ Il problema è dunque quello dell'identificazione degli «outliers»



37

Geometry Processing su point clouds: surface offsetting

Esempio di manipolazione di una point cloud:

- ✓ Task: surface offsetting:
 - aggiungere un piccolo «offset» ad una superficie,
 - ⇒ per es, per simulare l'aggiunta di un sottile strato di vernice sull'oggetto, avente un piccolo spessore, che cambia (sposta) la superficie
- ✓ Come ottenere questo risultato (idea): basta spostare la posizione di ogni punto della point-cloud lungo la sua normale, allontanandolo di un di un fattore scalare k (un parametro del metodo)
- ✓ Equazione (usando l'algebra di punti e vettori):

$$\forall \mathbf{p}_i \in P : \mathbf{p}_i \leftarrow \mathbf{p}_i + k \hat{\mathbf{n}}_i$$

- ✓ (Qual'è la complessità algoritmica di questo procedimento?)



38