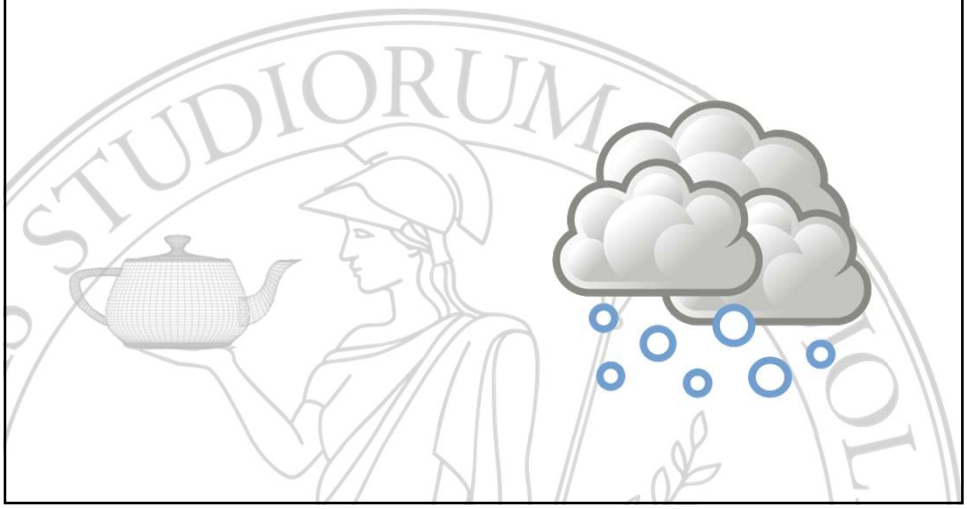
Marco Tarini - Computer Graphics 2026/2027
Università degli Studi di Milano

Point clouds (part 2)



17

Storage di point cloud

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

Posizione del primo campione (x y z)

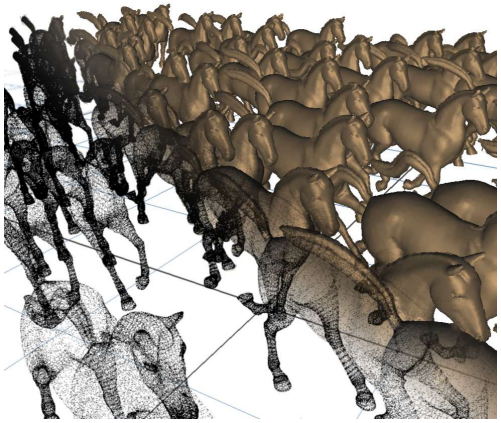
Normale del primo campione (x y z)

```
-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
```

19

Rendering di point cloud: point splatting

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

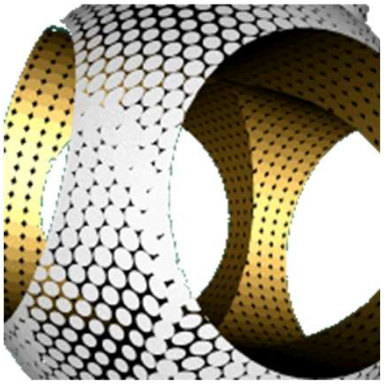


img by Michael Wimmer 2012

20

Rendering di point cloud: point splatting

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



img by Mario Botsch, 2004

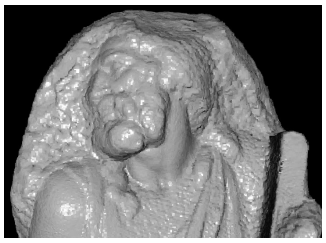
21

Multi-risoluzione su point-cloud

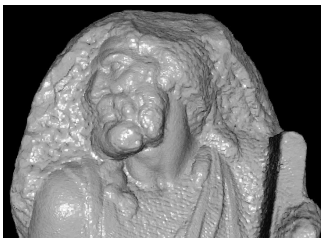
- ✓ La point-cloud è semplice da gestire perché ogni punto è indipendente dagli altri.
  - ⇒ Ad esempio…
- ✓ …una **multirisoluzione** si ottiene semplicemente prendendo un sottoinsieme arbitrario dei punti
- ✓ Possibile strategia: (adottata ad es. da «qSplat» - Stanford uni)
  - ⇒ preordinare la point cloud di N vertici in modo che i primi M<N vertici siano ben distribuiti, per ogni M
  - ⇒ Una permutazione casuale (*random shuffle*) è un'ottima approssimazione di questo!
  - ⇒ in fase di rendering, disegnare solo i primi M vertici
  - ⇒ la dimensione degli splat decresce al crescere del valore scelto per M

22

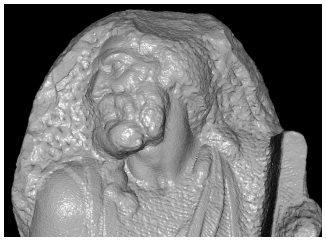
Multi-risoluzione su point-cloud



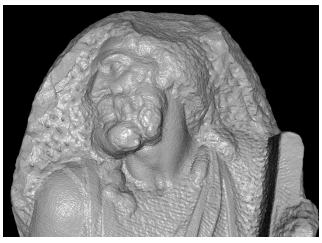
132,000 splats (132 ms)



260,000 splats (215 ms)



1,017,000 splats (722 ms)



14,836,000 splats (8308 ms)

Images by QSplat software

25

Acquisizione 3D di point-cloud

- ✓ Grazie alla loro semplicità, le point cloud si prestano a molte tecniche di aquisizione 3D
  - ⇒**Laser scanning**
  - ⇒**«Time of flight» scanner**
- ✓ In particolare, esistono tecniche per produrre una Point Cloud a partire da un insieme di immagini fotografiche
  - ⇒**Shape From Motion / Photogrammetry** (vediamo questa techinca, che è fra le più usate)

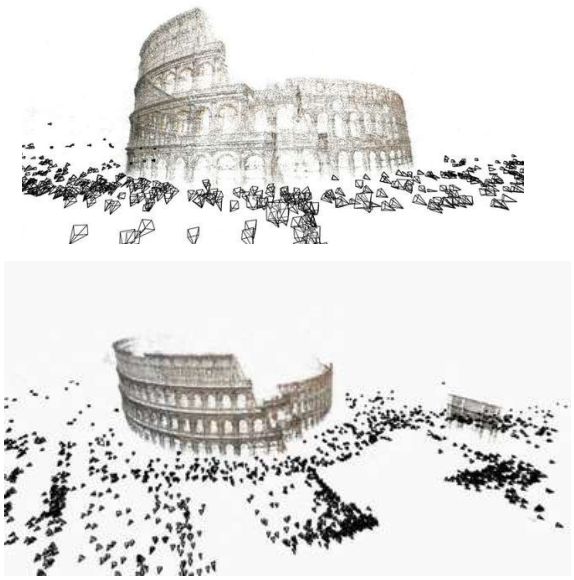
27

Tecniche fotogrammetriche: caratteristiche

- ✓ La nuvola di punti è ricostruita partendo semplicemente da un grande insieme di fotografie che rappresentano l'oggetto!
- ✓ Vantaggi:
  - ⇒ Bassi costi
  - ⇒ Nessun hardware specifcio è necessario per l'aquisizione (basta una comune macchina fotografica)
  - ⇒ Scale possibili: qualsiasi (da ambienti urbani a fotografie di microscopia)
- ✓ Svantaggi:
  - ⇒ Le nuvole di punti acquisite sono soggette in modo particolare ai difetti (rumore, incompletezza etc) che stiamo per vedere (e di cui vedremo anche alcune contromisure)
  - ⇒ E' possibile solo catturare oggetti relativamente opachi (sia nel senso di «non lucidi» che «non trasparenti»). Quindi non: vetro, superfici bagnate, metalli lucidi, specchi…
  - ⇒ E' una tecnica di acquisizione che, di base, costruisce solo nuvole di punti (il tipo strutturato di modelli 3D) – è di solito necessaria una conversione a modelli più strutturati, come vedremo

28

Una point-cloud acquisita con fotogrammetria



["Building Rome in a Day"
Agarwal et al, ICCV 2009]

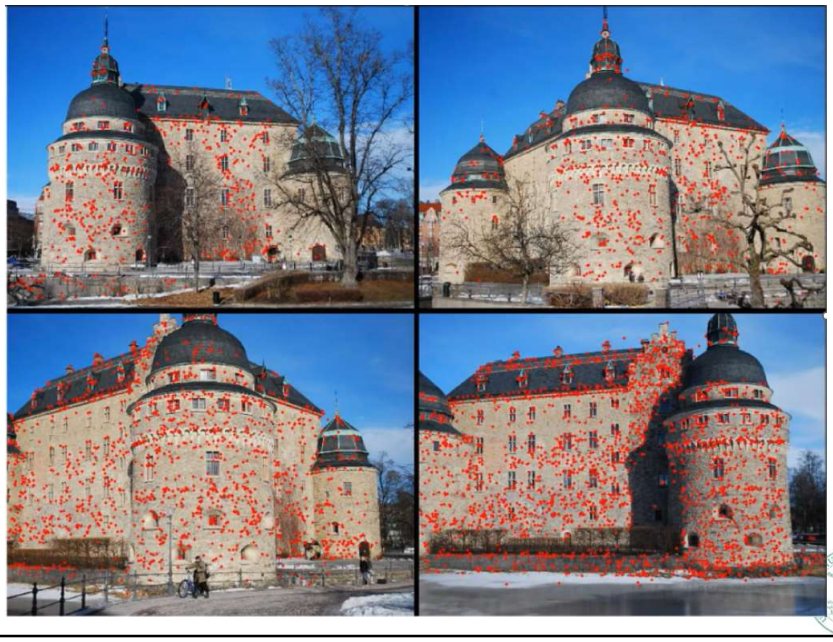
29

Acuisizione con fotogrammetria: le fasi

1. Cattura: scattare un gran numero di foto all'oggetto
   - ⇒ Possibilmente, in condizioni di luce buone e costanti
   - ⇒ E' necessario inquadrare ogni punto della superficie da punti di vista diversi (altrimenti, non saranno ricostruite)
2. Per ogni immagine: identificare molti *feature point*
   - ⇒ Dettagli sulla superficie che potranno essere facilmente riconosciuti in altre immagini
   - ⇒ Ad esempio, non zone con un colore uniforme
3. Su tutte le immagini: match dei feature point
   - ⇒ Quale feature point nell'immagine A è quale dell'immagine B?
4. Fase globale: ricostruire simultaneamente…
   - ⇒ i punti di vista delle immagini (posizioni delle macchine fotografiche), e
   - ⇒ le posizioni 3D di punti sulla superficie
5. …in modo che le proiezioni di ogni punto 3D su ogni immagine corrisponda a matching feature point

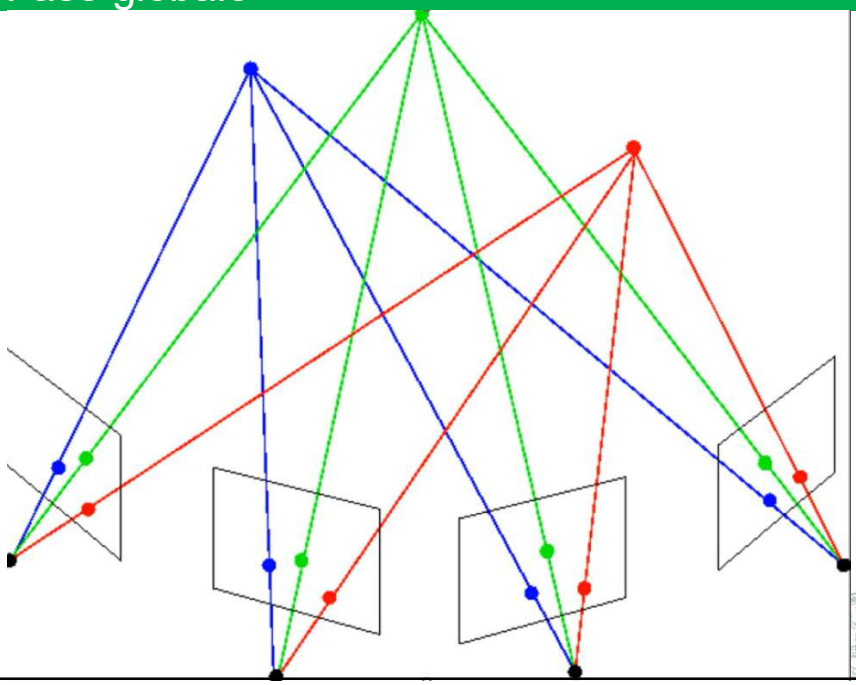
31

2. Identificazione dei feature points



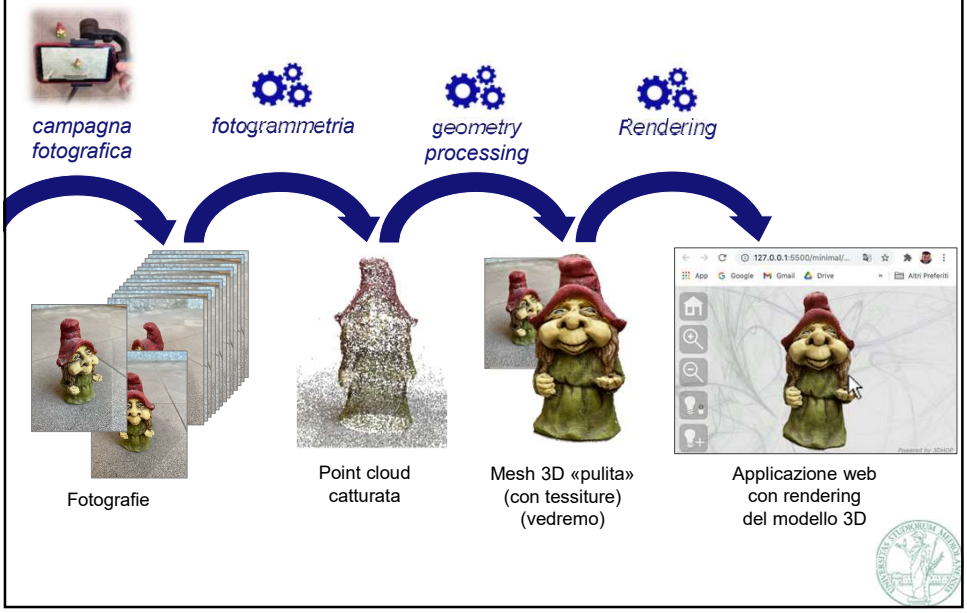
32

4. Fase globale



33

Un esempio di pipeline da foto a modello 3D su web



36

Acquisizione di point cloud con fotogrammetria

37

Alcuni software e strumenti esistenti

✓ Software specializzato per fotogrammetria:

Meshroom

Regard3D

Metashape

MicMac

3DF ZEPHYR

COLMAP

✓ Repository pubblici di point-cloud acquisite da modelli reali:

Sketchfab

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

Real-World Textured Things

https://texturedmesh.isti.cnr.it/

(nota: sono processate ed esposte come mesh – vedi prossime lezioni)

38

Alcuni software / strumenti Open-Source esistenti

Per geometry-processing effettuato su point-clouds

pcl

Point Cloud
(libreria C++)

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

MeshLab
(applicativo software)

Specializzato sorpattutto su mesh poligonali, ma lavora anche su point-clouds. Vedi: «Filters» → «Point Set»

Cloud-Compare
(applicativo software)

Specializzato sorpattutto su point-cloud

39

Point cloud acquisite: tipiche limitazioni

- ✓ Campionamento inadeguato
  - ⇒ Eccessivo (risoluzione eccessiva per gli scopi)
  - ⇒ Insufficiente (risoluzione insufficiente per rappresentare i dettagli)
  - ⇒ Poco uniforme (in modo non controllato).
    In alc
- ✓ Rumore
  - ⇒ nelle posizioni
  - ⇒ nella normale
- ✓ Presenza di «**outlier**»
  - ⇒ punti spuri che non appartengono alla superficie di interesse, o a nessuna superficie
- ✓ Incompletezza: le superfici sono incomplete
  - ⇒ Perché, ad esempio (usando fotogrammetria)
    una parte della sup non è visibile da un numero sufficiente di fotografie, e da alcuna fotografia
  - ⇒ Oppure perché le superifici trasperanti o lucide (presenza di riflessi) rompono le assunzioni della fotgrammetria

40

Geometry Processing su point-clouds (sommario)

Alcuni dei procedimenti comuni che lavorano su point cloud:

- ✓ *Clean up*, cioè ridurre in post-processing (post-acquisizione) di alcuni dei problemi delle point cloud acquisite
  - ⇒ Rimozione / riduzione rumore
  - ⇒ Rimozione degli *outlier*
  - ⇒ Completamento delle parti mancanti
  - ⇒ Rendere la risoluzione maggiore, oppure minore, oppure più uniforme
- ✓ Ricostruzione normali dalle posizioni
  - ⇒ qualora non presenti.
    Ad esempio, la fotogrammetria tipicamente non stima le normali
- ✓ Registrazione / allienamento
  - ⇒ (vedi in seguito)
- ✓ *Surface reconstruction*: conversione di questo tipo di dato in strutture dati più utilizzabili, tipicamente mesh poligonali
  - ⇒ vedi lezione successive

Vedremo prima un sottoproblema ricorrente per quasi tutti questi task: la definizione di un rapporto di vicinato fra i vertici della point-cloud.

41

Geometry Processing su point clouds: un sotto task

- ✓ Un sotto-problema ricorrente in molti task:
  trovare il **vicinato spaziale** di ogni vertice
  - ⇒ Cioè rispondere alla domanda:
    dato un vertice (che è un punto sulla superficie), quali sono i vertici che fanno parte dell'intorno della superficie passante per quel punto?
- ✓ Una naturale definizione geometrica 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
  - ⇒ esercizio: esprimi questa condizione geometrica con un'espressione dell'algebra di punti e vettori vista
- ✓ Questa caratterizzazione del problema ha un difetto
  - ⇒ è molto difficile determinare un buon valore per d_{MAX}
  - ⇒ dipende dalla scala (è ***scale dependent***) e dalla risoluzione del modello
  - ⇒ nessun valore (costante) funziona bene quando la **risoluzione** è **adattiva** o cmq non costante.

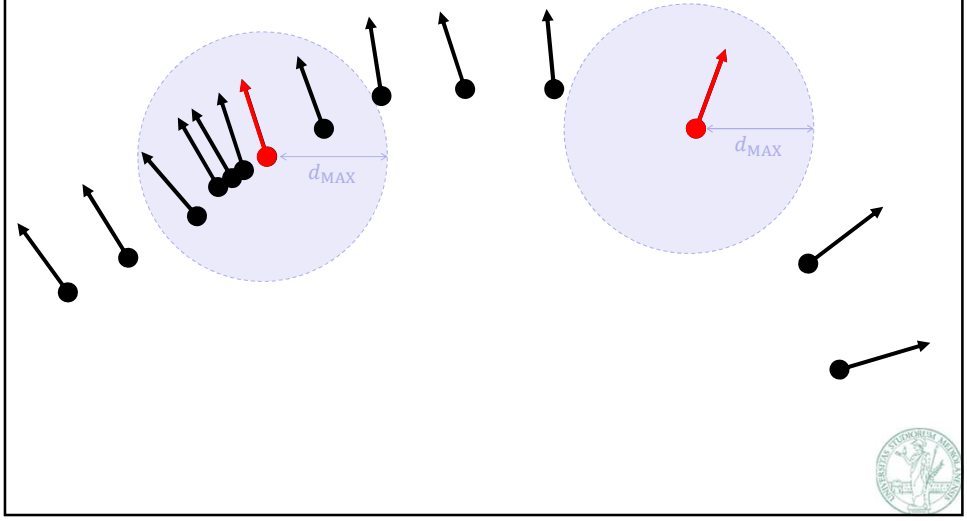
42

Geometry Processing su point clouds: k-NN

- ✓ Una *migliore* ri-definizione del problema:
  - ⇒ «dato un vertice i in posizione $\mathbf{p}_i$,
    trovare i k vertici più vicini a lui»
  - ⇒ dove k è una 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
  - ⇒ 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)
    Quindi: trovare il vicinato di tutti i punti: complessità quadratica
  - ⇒ (vedi corso di algoritmi per la spiegazione di questi termini!)
  - ⇒ INFO: in CG e in Geometry processing, una complessità quadratica è tipicamente considerata proibitiva
  - ⇒ Sono necessarie tecniche algoritmiche più sofisticate (che non vedremo)

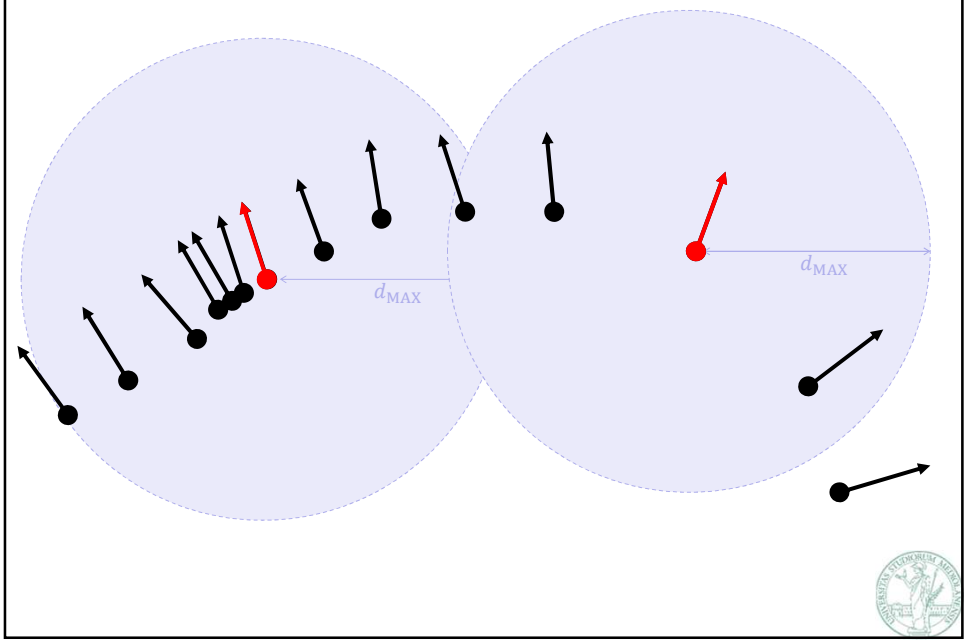
43

Definzione del vicianto



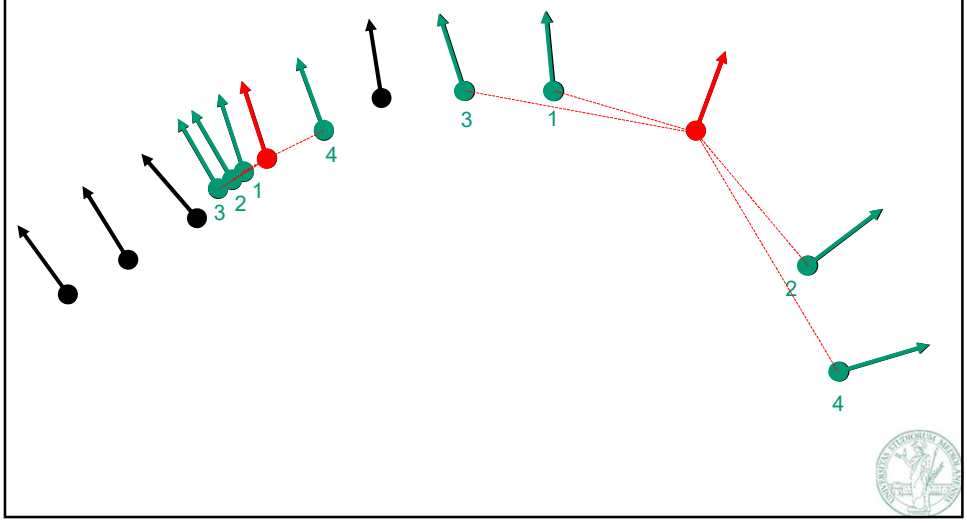
44

Definzione del vicianto



45

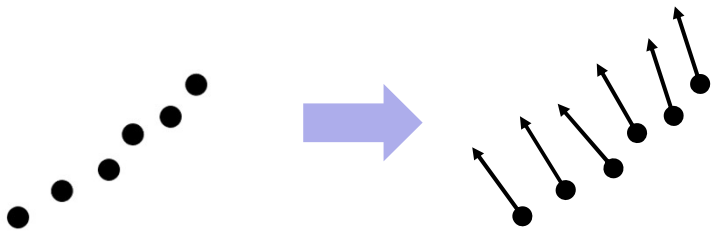
Definzione del vicianto: k-NN (qui, con k = 4)



47

Geometry Processing su point clouds: stima delle normali

✓Stima delle normali su una point-cloud



48

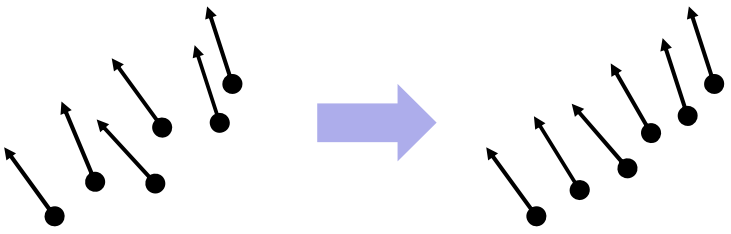
Geometry Processing su point clouds: stima delle normali

- ✓ Dipendendo dalla tecnica di generazione, alcune point-cloud non partono provviste di direzioni normali per ogni campione
- ✓ E' dunque necessario stimare le normali di ogni vertice (a partire dalle posizioni dei vertici)
- ✓ Come? (schema di una soluzione)
  - ⇒ Per ogni vertice i:
    1. trovare il suo vicinato (il suo ***k*-NN**)
    2. trovare il piano passante per il suo vicinato («best fitting plane»)
    3. assegnare al vertice i la normale di quel piano
  - ⇒ Nota: per tre punti passa un piano, ma nessun piano in generale passa per N>3 punti. Tuttavia, è possibile trovare il piano che *minimizza la distanza da N punti dati* (un problema da formulare e risolvere, come sempre, usando l'algebra di punti e vettori)
  - ⇒ Nota: è necessario risolvere anche un'ambiguità di direzione: un piano ammette due vettori normali di verso opposto, e ne va scelto uno (in modo consistente)

49

Geometry Processing su point clouds: denoising

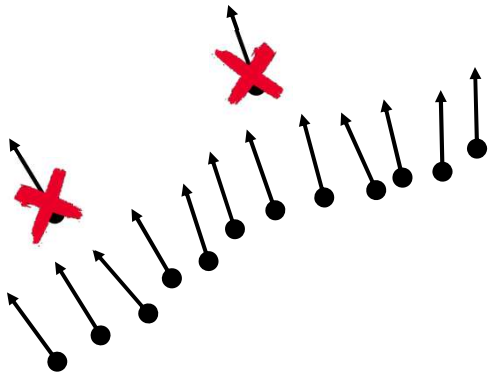
- ✓ Denoise – Noise reduction – o «fairing»
  - ⇒ Di posizioni (e/o normali)



50

Geometry Processing su point clouds: outlier removal

- ✓ Rimozione degli «outlier»



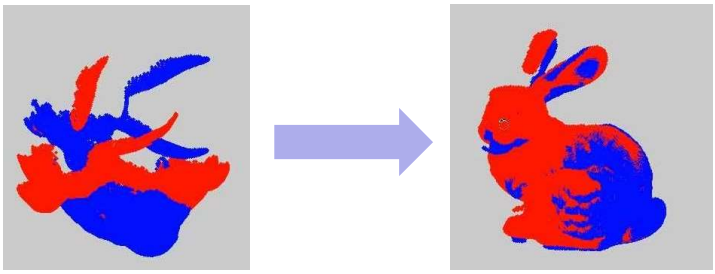
51

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 errore, o perché sono relativi ad altre superfici): questo punti non devono essere *rimossi*, non modificati
  - ⇒ Il problema è dunque quello dell'identificazione degli «outliers»

52

Geometry Processing su point clouds: registration



- ✓ Spesso due o più point-cloud descrivono parti diverse di uno *stesso* oggetto
  - ⇒ ad esempio, quando ho catturato due lati di una statua con due acquisizioni automatiche (come laser scanning), ottenendo due nuvole separate
  - ⇒ Ogni point cloud è espressa in un suo proprio Sistema di riferimento
  - ⇒ Per poter fondere le due nuvole in una sola (più completa!) è necessario prima portarle nello stesso Sistema di riferimento
- ✓ In pratica si tratta di riposizionare (ruotare e traslare) una delle due in modo da portare *le parti in comune* a (quasi) coincidere
  - ⇒ nota: quindi, ci devono essere quindi sovrapposizioni!
  - ⇒ è problema detto “allineamento” o “registrazione” (di point cloud)

55

Geometry Processing su point clouds: registration

Distinguiamo due fasi (in cascata)

- ✓ **Coarse registration** (allineamento iniziale):
  - ⇒ trovare una prima soluzione che metta in corrispondenza le due point-cloud in modo approssimato
  - ⇒ Sono stati proposti molti algoritmi automatici (che non vedremo),
  - ⇒ Alcune soluzioni utilizzate richiedono un intervento da parte di un operatore umano
- ✓ **Fine registration** (allineamento fine):
  - ⇒ una volta che le parti comuni delle due superfici sono in reciproca prossimità, l’allineamento iniziale viene reso molto più accurato da apposti algoritmi.
  - ⇒ Vediamo una classe di questi algoritmi

60

Geometry Processing su point clouds: registration

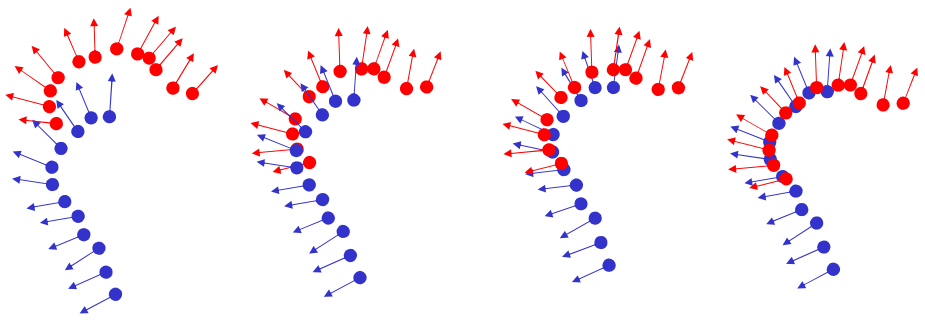
- ✓ Schema di una soluzione del problema dell'allineamento fine: algoritmo "Iterative Closest Points" (**ICP**)
  1. Scegliere un insieme arbitrario di punti di una delle due nuvole (per esempio, qualche centinaio)
  2. Per ognuno di loro, trovare il suo *corrispondete* sull'altra nuvola, come il punto più vicino (closest) ad esso
  3. Scartare i punti che hanno il corrispondente troppo lontano (oltre ad un valore soglia)
  4. Trovare una rotazione+traslazione che avvicini il più possible ogni punto sul suo corrispondete (un sotto problema di natura matematica)
  5. Ripetere (cioè *iterare*) dal punto 1, fino a "convergenza" (cioè, fino a che non ci sono più miglioramenti sostanziali)

61

Geometry Processing su point clouds: registration

Allineare una point cloud su un'altra

- ✓ Algoritmo ICP («Iterative Closest Points»)



situazione iniziale | dopo la 1° iterazione | dopo la 2° iterazione | dopo la 3° iterazione (e convergenza)

62

Geometry Processing su point clouds: registration

- ✓ Caratteristiche dell'ICP
  - ⇒ Ad ogni iterazione, le due nuvole si avvicinano e quindi si avranno corrispondenze migliori nell'iterazione successiva, e così via
  - ⇒ Se nella situazione di partenza le due nuvole sono sufficientemente vicine alla soluzione corretta, allora il sistema converge ad una soluzione molto buona («incollando» le due point clouds in modo perfetto)
  - ⇒ Altrimenti, il sistema può convergere a soluzioni del tutto sbagliate, o non convergere del tutto
  - ⇒ E' necessario partire da una soluzione che si avvicina a quella «corretta». Questo è il compito della fase coarse dell'allineamento

63

Geometry Processing su point clouds

- ✓ Un task comune: convertire la point-cloud in un'altra rappresentazione
  - ⇒ Più adatta a processing e/o a rendering
  - ⇒ Ricorda che le point cloud sono diffuse soprattutto in virtù della loro facilità di acquisizione 3D (compreso da immagini: fotogrammetria) non per la loro praticità di uso
- ✓ Esempio tipico: da point cloud a mesh poligonale
  - ⇒ Lo vedremo nelle prossime lezioni!

65