Extending IFC with point cloud data

eg-ice 2015

Thomas Krijnen, Jakob Beetz
Eindhoven University of Technology, The Netherlands

Sebastian Ochmann, Richard Vock, Raoul Wessel
Institute of Computer Science II, University of Bonn, Germany
Aims

Unite both data carriers
In a harmonized storage format
By extending the IFC schema

Introduce a binary serialization format for IFC
Aims
IFC and point clouds anno 2014

#1=IFCCARTESIANPOINTLIST3D((121.76123809814453,-78.45201110839844,-13.620319366455078),
(121.8034133911328,-78.57342529296875,-12.835342407226562), (121.7803762196289,-78.34296417236328,-13.846855163574219), (121.78942108154297,-78.3430404663086,-13.734870910644531), (121.7864288330078,-78.33686065673828,-13.62184906005894), (121.79190826416016,-78.33324432373047,-13.50927734375), (121.80005645751953,-78.33274841308594,-13.397228240966797), (121.82268524169922,-78.3357925415039,-13.173976707421875), (122.14295196533203,-78.85536193847656,-17.720565795898438), (122.14188385009766,-78.8489990234375,1.8842010498046875), (122.1812515258789,-78.74574279785156,1.8790130615234375),
(2.644031524658203,-14.406140327453613,1.141799267578125), (2.628744125366211,-14.345318794250488,1.1577377319335938), (3.482269287109375,17.248559951782227,-1.4286651611328125),
(3.32574462890625,-17.08686447135547,-1.4688568115234375), (3.4234485626220703,-15.845314025878906,-1.2468643188476562), (3.449533462524414,-15.883966445922852,-1.21118640625), (3.5097713470458984,-15.5851202011084,-1.1186370849609375), (4.405935287475586,-14.289386749267578,-1.1261444091796875),
(4.405914306640625,-14.280056953430176,-1.043464606445312), (4.378328323364258,-14.243566513061523,-1.0649948120117188), (5.349382400512695,-14.975531578063965,-0.90240478515625), (5.226930618286133,-14.915581703186035,-0.9336929321289062), (2.618467330932617,-10.69272427905273,-1.2475967407226562), (2.618467330932617,-10.69272427905273,-1.2475967407226562),
(2.516057968139484,-10.395642280578613,-1.3473358154296875), (2.638456344604492,-10.4812889099121,-1.3255081176757812), (2.763673782348633,-10.58719539642334,-1.269317626953125),
(2.400505659179688,-10.004755973815918,-1.356649169921875), (2.838045196533203,-10.220603942871094,-1.3677139282226562), (2.854219436645058,-10.277368545532227,-1.3037109375), (2.8745193481445312,-10.280534744262695,-1.2966690063476562), (2.851787567138672,-10.229779243649238,-1.3521499633789062),
(2.974489212036133,-10.0681791305542,-1.234466552734375), (1.8945045471191406,-10.313359260559082)
Points for improvement:
No per-point attributes, such as colors, etc.
No level of detail
No explicit means for decomposition
No metadata, such as scanner model, etc.
No way to extract localized subsets
Slow to parse and leading to exorbitant data sizes
Proposed structure

A schema extension with point cloud compression techniques that deepens the semantic relation between points and building elements.
Proposed structure

Three layers of compression:
- Project points into **bounded parametric space**
- Reduce the dimensionality of points by **grid discretization**
- **Transparent compression** on a file format level (HDF5)
Proposed structure

Cartesian coordinates offer:

An *easy and intuitive* way to store point cloud data.

Intended for *geographic features* or *unassociated points* unrelated to building element.
Proposed structure

Parametrization

Associated points described in the **parametric** space of building surfaces.

Three components \{u,v,w\} required, but the range of values is **bounded** and can be more efficiently encoded.
Proposed structure

Height fields

{u,v} components defined as the intersection of grid axes.
Per point only one component needs to be stored.
Proposed structure

Floating point discretization

(0.8482145585275755, 0.4089384818729891, 0.8027061702482456, 0.11449717768247669)
Proposed structure

Floating point discretization

\[(0.8482145585275755, 0.4089384818729891, 0.8027061702482456, 0.11449717768247669)\]

\[\approx \]

\[
\begin{array}{cccc}
(55587, & 26799, & 52605, & 7503) \\
65535 & 65535 & 65535 & 65535 \\
\end{array}
\]
**Proposed structure**

**Floating point discretization**

\[
(0.8482145585275755, 0.4089384818729891, 0.8027061702482456, 0.11449717768247669)
\]

\[
\begin{array}{cccc}
\approx & \approx & \approx & \approx \\
(55587, & 26799, & 52605, & 7503) \\
65535 & 65535 & 65535 & 65535 \\
\end{array}
\]

\[
((55587, 26799, 52605, 7503), 65535)
\]
Proposed structure

Floating point discretization

\[(0.8482145585275755, 0.4089384818729891, 0.8027061702482456, 0.11449717768247669)\]

\[\approx\]

\[(55587, 26799, 52605, 7503)\]

\[\approx\]

\[65535\]

\[65535\]

\[65535\]

\[65535\]

\[\approx\]

\[\approx\]

\[\approx\]

\[\approx\]

\[\approx\]

\[65535\]

\[(55587, 26799, 52605, 7503), 65536\]
Proposed structure
Proposed structure
Proposed structure
Proposed structure
Serialization in HDF5

HDF5 is a:
- hierarchical, binary data format
- for heterogeneous data
- stored according to self-documenting data types
- with b-tree indexes and transparent compression.
- It is an open standard and an EXPRESS-HDF5 mapping is standardized as ISO 10303-26.
Serialization in HDF5
Serialization in HDF5
Serialization in HDF5
A prototypical implementation is provided using only open source components, such as:

- IfcOpenShell,
- h5py,
- Point Cloud Library (PCL),
- e57lib,
- NumPy,
- Scipy and
- OpenCascade
Results

The dataset for this experiment consists of a professionally modelled IFC file and subsampled point cloud scan of roughly 15 million points.
Results
Results
Points for improvement:
No per-point attributes, such as colors, etc.
No level of detail
No explicit means for decomposition
No metadata, such as scanner model, etc.
No way to extract localized subsets
Slow to parse and leading to exorbitant data sizes
Points of improvement:

- No per-point attributes, such as colors, etc.
- No level of detail
- No explicit means for decomposition
- No metadata, such as scanner model, etc.
- No way to extract localized subsets
- Slow to parse and leading to exorbitant data sizes

Efficient to parse and file sizes under industry standard point cloud formats.
IFC and point clouds anno 2015
IFC and point clouds anno 2015

Extending IFC with point cloud data

eg-ice 2015

Thomas Krijnen, Jakob Beetz
Eindhoven University of Technology, The Netherlands

Sebastian Ochmann, Richard Vock, Raoul Wessel
Institute of Computer Science II, University of Bonn, Germany

TU/e
Technische Universiteit Eindhoven
University of Technology

Where innovation starts