Digital Humanities im Museum 4.0

Erstellung

Julia Albers, Jannick Bock, Felix Harms, Luca Junge, Pepijn Kampschöer,
Leon Leymann, Christian Lüth, Björn Schultz, Fabian Sehm, André Stadtelmeyer

https://informatik.hs-bremerhaven.de/dh2018/

Multi-View Environment

Die Multi-View Environment (MVE) bietet eine vollständige Pipeline zur Photogrammetrie. Diese umfasst Structure-from-Motion, Multi-View Stereo und Surface Reconstruction. Die Pipeline besteht aus einzelnen Bibliotheken, welche in C++ geschrieben sind. Der Quelltext ist plattformunabhängig und läuft unter Linux, MacOS als auch Windows. [1]

MVE wurde an der Technischen Universität Darmstadt durch die Forschungsgruppe "Graphics, Capture and Massively Parallel Computing" entwickelt. Leiter der Forschungsgruppe ist Prof. Dr.-Ing. Michael Goesele. Zusätzlich haben an diesem Projekt Simon Fuhrmann, Fabian Langguth, Nils Moehrle und Michael Waechter mit gearbeitet.

Ein großer Vorteil für das Projekt "Digital Humanities im Museum 4.0" war die Nutzbarkeit von MVE in der Kommandozeile. Dadurch konnte ein Skript erstellt werden, welches die Einzelschritte der Rekonstruktion automatisch durchführt.

```
#!/bin/bash
if [[ -d $1 ]]; then
  time(
    sourcePath=$1
    workingPath=$sourcePath"working/"
    projectName=${sourcePath:0:-1}
    projectName=${projectName##*/}
    originalPly="$workingPath$projectName.ply"
    structurePly="$workingPath${projectName}_structure.ply"
    cleanPly="$workingPath${projectName}_clean.ply"
    /opt/mve/apps/makescene/makescene -i $sourcePath $workingPath
    /opt/mve/apps/sfmrecon/sfmrecon $workingPath
    /opt/mve/apps/dmrecon/dmrecon $workingPath
    if [[ -d ${workingPath}views/view_0000.mve/ ]]; then
      depth=$(ls ${workingPath}views/view_0000.mve/depth*)
      depth=${depth:0:-5}
      depth=${depth##*pth-L}
      /opt/mve/apps/scene2pset/scene2pset -F $depth $workingPath
  $originalPly
      /opt/mve/apps/fssrecon/fssrecon $originalPly $structurePly
      /opt/mve/apps/meshclean/meshclean $structurePly $cleanPly
    else
      echo "Fehler bei der Erstellung der Punktwolke"
    fi
  )
else
    echo -e "$1 is not valid\nimg2threed [Image Source Directory]"
    exit 1
fi
```

Listing 1: Quelltext Bash Skript img2threed.sh

COLMAP

COLMAP ist ebenfalls eine Pipeline zur 3D-Rekonstruktion aus Fotos. Die Pipeline umfasst Structure from Motion [2] und Multi-View Stereo. [3] Sie wurde an der Eidgenössischen Technischen Universität Zürich und der University of North Carolina at Chapel Hill entwickelt. Hauptentwickler ist Johannes L. Schönberger, außer ihm haben noch Jan-Michael Frahm, Marc Pollefeys und Enliang Zheng bei der Entwicklung mitgewirkt.

Das Ziel des Projektes COLMAP war es, einen neuen Ansatz für Photogrammetrie zu finden:

> Es sollten nicht Objekte aus einzelnen Fotoserien rekonstruiert werden, viel mehr wollte man sich die Macht der Masse zu nutzen machen.

Das bedeutet viele einzelne Fotos aus dem Internet zu nutzen, welche von verschiedenen Personen mit verschiedenen Geräten, zu unterschiedlichen Zeiten gemacht wurden. Dieser Ansatz zielt nicht auf kleine Einzelobjekte ab, hierbei geht es viel mehr um eine großflächige Rekonstruktion z.B. von Gebäuden oder ganzen Stadtteilen.

Das imposanteste Beispiel hierfür ist die Erstellung einer Punktwolke von der Innenstadt von Rom. Aus 74.394 Bildern hat die Software 20.918 Bilder einander zuordnen können und 5,3 Millionen Punkte angeordnet.

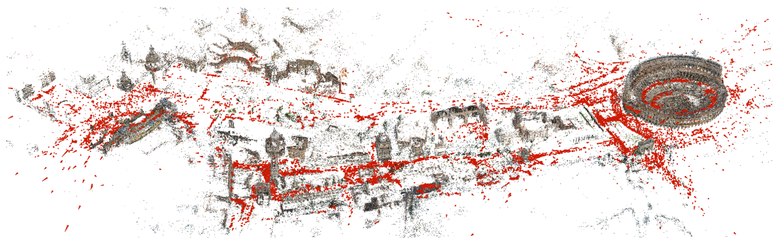

Abbildung 1: Punktwolke vom Roms Zentrum, erstellt mit COLMAPs SfM Pipeline. [2]

AliceVision

AliceVision ist ein quelloffenes Gemeinschaftsprojekt, an dem sich Vertreter aus der Industrie, wie auch aus dem akademischen Bereich beteiligen. [4] Das Projekt ist in sechs Teilprojekte aufgeteilt:

1. AliceVision
2. CCTag (C++)
3. Meshroom (Python)
4. MeshroomMaya (C++)
5. Popsift (Cuda)
6. QtOIIO (C++)

Die Basis des Frameworks, welches die Funktionalität der Photogrammetrie bietet, ist das Teilprojekt AliceVision. Das Framework selbst ist modular aufgebaut, d.h. einzelne Funktionen wie „Feature Extraction“ sind einzelne Programme, denen die nötige Konfiguration über Aufrufparameter übergeben werden. Dies hat den Vorteil, dass jede Funktionalität vom Framework individuell von anderen Programmen konsumiert werden können. [5]

Ein Beispiel hierfür ist das Teilprojekt „Meshroom“. Meshroom nutzt die von AliceVision bereitgestellten Programme, um eine Benutzeroberfläche zur Verfügung zu stellen. [6] Die Benutzeroberfläche ist darauf ausgelegt die Pipe zu konfigurieren oder ggf. zu erweitern, Bilder für die Photogrammetrie entgegenzunehmen und die Zwischen- und Endergebnisse zwischen den einzelnen Programmaufrufen darzustellen. Hervorzuheben ist, dass Meshroom in einer anderen Programmiersprache geschrieben wurde als AliceVision. Durch die modulare Architektur ist es ohne Probleme möglich das Projekt mit anderen Technologien zu erweitern.

Meshoptimierung

Bei der Erstellung von Meshes über den photogrammetrischen Prozess entstehen sehr komplexe Punktwolken. Diese erfordern viel Speicherplatz und bei der Darstellung mehr Rechenleistung. Ein Teilziel dieses Projekts war es die Meshes mit Webtechnologien im Browser darzustellen. Um Ladezeiten und Speicherauslastung zu verringern müssen die Punktwolken vereinfacht werden. Für die Vereinfachung müssen Punkte aus der Wolke so entfernt werden, das die Oberfläche des Meshes sich im geringst möglichen Maße verändert.

Dafür gibt es verschiedene Algorithmen zum Beispiel die Kanten-Kollaps-Transformation, die Clustering Dezimierung und die Quadric Edge Collapse Dezimierung. Im Projekt wurde mit dem letzten genannten Algorithmus experimentiert. Bei diesem konnten selbst bei einer Reduzierung der Punktwolke um bis zu 80 Prozent, noch Ergebnisse erzielt werden bei denen das Ergebnismesh dem Ursprungsmesh qualitativ sehr ähnlich sah. Dadurch ergaben sich Speicherersparnisse von bis zu 75 Prozent.

Konvertierung und Website-Generierung

Die berechneten Modelle werden über die Verwendung eines Konvertierungsskriptes in das gewünschte Format gebracht. Die Pipelines erzeugen unterschiedliche Dateiformate:

- AliceVision, "WAVEFRONT Object" (.obj)
- COLMAP, "Polygon File Format" (.ply)
- MVE, "Polygon File Format" (.ply)

Beides sind gängige offene Formate zur Speicherung von dreidimensionalen geometrischen Formen.

Mithilfe des Programmes Meshlabserver [7] wird das Polygon File Format in das .obj-Format konvertiert.

Dieser Zwischenschritt ist notwendig, da das bevorzugte 3D-Format im Projekt "Digital Humanities im Museum 4.0" das GL Transmission Format (.gltf) ist. Das .obj-Format ist bereits ein sehr speicheroptimiertes Format und lässt sich durch eine einfache Konvertierung durch das Programm "obj2gltf" [8] zum gewünschten .gltf-Format formatieren. Diese Umwandlung kann zusätzlich über das Open Source 3D-Programm Blender erfolgen. [9] Blender bietet den Vorteil, dass sich zusätzlich Änderungen am Modell vornehmen lassen.

Die GLTF Datenstruktur wurde von der Khronos Group entwickelt, dieses Format ermöglicht eine starke Komprimierung des Modells und optimiert dazu noch die Ressourcenauslastung durch die Nutzung von WebGL. [10] Die Umwandlung erlaubt es, die endgültigen Modelle speichergünstig, in das eigens erstellte Template einer three.js Szene einzubetten.

Abschließend wird die erzeugte Website automatisch auf der projekteigenen Website hochgeladen.

Einsatz von three.js im Projekt

Bei three.js handelt es sich um eine browserübergreifende Open Source JavaScript-Bibliothek um 3D-Grafiken im Webbrowser anzuzeigen. Die Darstellung von 3D-Modellen ist ein zentraler Aspekt des Projekts. Um beeindruckende Ergebnisse bei der Darstellung unserer mit Photogrammetrie erstellten 3D-Modelle zu erzielen, bietet three.js uns mit einfachen Mitteln die Möglichkeit dazu. Wir haben mit verschiedenen Effekten und Interaktionsmöglichkeiten experimentiert. Besonderen Wert haben wir dabei auf die möglichst realistische Darstellung gelegt. Geholfen haben uns dabei detaillierte Einstellungsmöglichkeiten bei Licht, Schatten und Kamerafahrten. Die genau Beschreibung des Einsatzes von three.js im Projekt ist auf dem Plakat "Präsentation" zu finden.

Literatur

[1] Simon Fuhrmann, Fabian L. ; Goesele, Michael: MVE – A Multi-View Reconstruction Environment. In: *Eurographics Workshop on Graphics and Cultural Heritage*, 2014

[2] Schönberger, Johannes L. ; Frahm, Jan-Michael: Structure-from-Motion Revisited. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016

[3] Schönberger, Johannes L. ; Zheng, Enliang ; Pollefeys, Marc ; Frahm, Jan-Michael: Pixelwise View Selection for Unstructured Multi-View Stereo. In: *European Conference on Computer Vision (ECCV)*, 2016

[4] AliceVision: *About*. https://alicevision.github.io/#about. Version: 2018. – abgerufen am 17. Dezember 2018

[5] AliceVision: *Photogrammetry Pipeline*. https://alicevision.github.io/#photogrammetry. Version: 2018. – abgerufen am 17. Dezember 2018

[6] AliceVision: *Meshroom*. https://alicevision.github.io/#meshroom. Version: 2018. – abgerufen am 17. Dezember 2018

[7] Visual Computing Lab: *Meshlab Server Git Repository*. https://github.com/cnr-isti-vclab/meshlab/tree/master/src/meshlabserver. Version: 2018. – abgerufen am 27. Dezember 2018

[8] AnalyticalGraphicsInc: *obj2gltf*. https://github.com/AnalyticalGraphicsInc/obj2gltf. Version: 2018. – abgerufen am 10. Januar 2019

[9] Blender Foundation: *Blender*. https://www.blender.org/. Version: 2019. – abgerufen am 04. Januar 2019

[10] Khronos Group: *GL Transmission Format*. https://www.khronos.org/gltf/. Version: 2019. – abgerufen am 07. Januar 2019