opencalibration

with Julian Kent

Zurich C++ Meetup

Agenda

- What is photogrammetry
- Typical photogrammetry pipeline
- What makes `opencalibration` different
- What’s next for `opencalibration`
- Demo

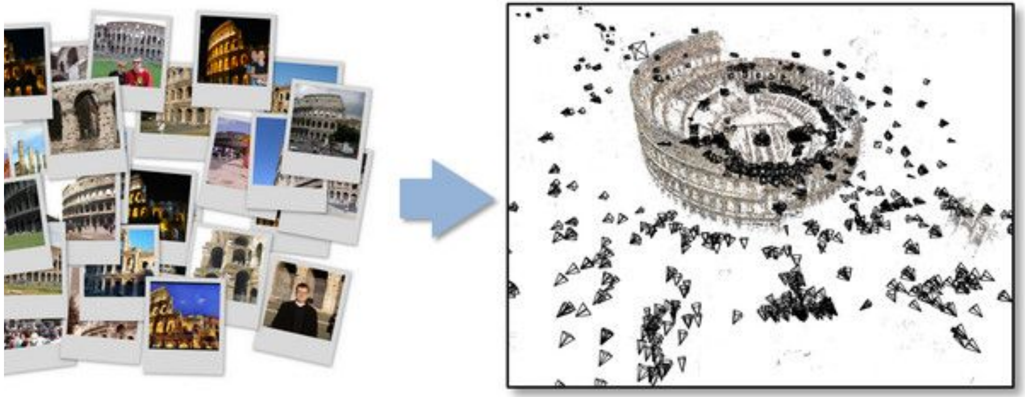
Photogrammetry

- Images → Camera model + sparse 3D point cloud
- Camera model: lens distortions + global position / orientation

Typical Photogrammetry Pipeline

Bundler

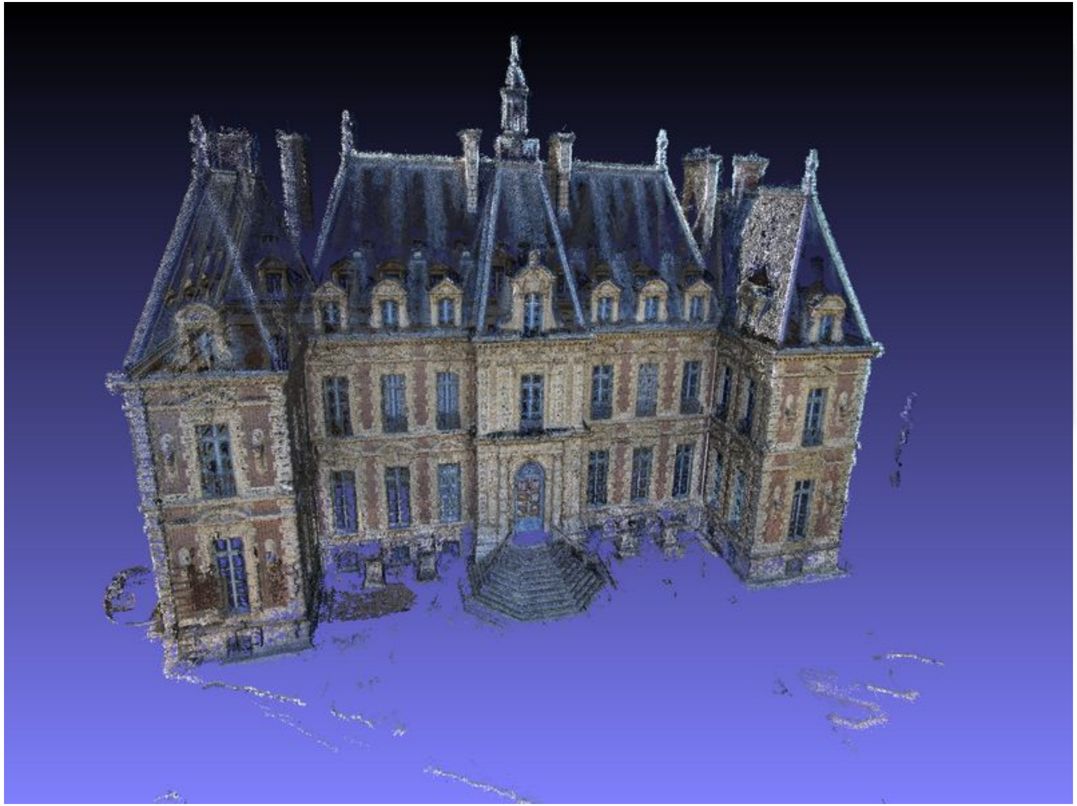
- Image loading
- Feature Extraction
- Matching
- RANSAC
- Decomposition, or PnP
- Bundle Adjustment



Typical Photogrammetry Pipeline

Bundler

- Image loading
- Feature Extraction
- Matching
- RANSAC
- PnP
- Bundle Adjustment



The `opencalibration` pipeline

- Load and initial alignment
- Camera pose & lens parameter refinement
- Low-res surface model estimation
- DSM + ortho preview (in progress)
  - Thumbnails from input images are used to quickly generate a low-resolution orthomosaic
  - Also generate a DSM to go with it

The `opencalibration` pipeline

Load and initial alignment

- Works ‘online’, new data can be added while processing

Split into different pipelined stages, on batches of images

Stage 1:

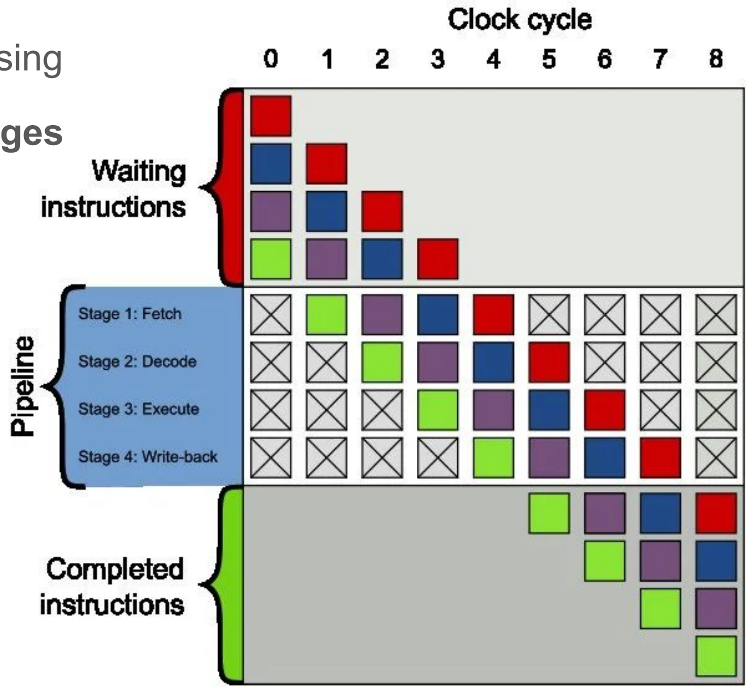
- Image Loading & Feature Extraction
  - IO and FPU intensive, uses lots of memory:

Stage 2:

- Matching, RANSAC, Decomposition
  - POPCNT heavy / $O(N^2)$ on features / future GPU offload:

Stage 3:

- Camera orientation optimization with local cluster
  - Lots of small serial operations, single threaded



The `opencalibration` pipeline

Camera pose & lens parameter refinement

- Mostly a ‘standard, classic’ bundle adjustment

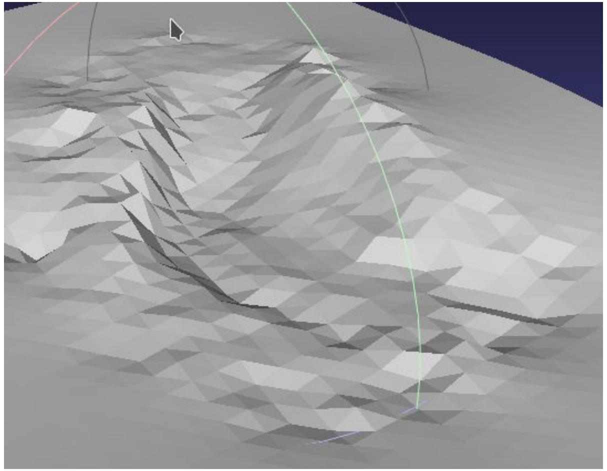
- What’s special:
  - Split images based on Spectral Clustering of measurement graph
  - Alternate between optimizing camera pose and lens parameters
  - Do pose graph minimization on each cluster separately - use a low-resolution surface mesh instead of individual keypoints.
    - Avoid O(N^3) scaling by breaking into clusters and parallelizing
  - Estimate camera parameters with classic keypoints-style bundle adjustment
    - Avoid O(N^3) scaling by only estimating camera parameters on largest cluster

The `opencalibration` pipeline

Surface model estimation

- Low resolution surface mesh directly optimized in clusters
- Meshes merged and final global optimization at the end - $O(N^3)$

opencalibration - 2m45s on a laptop



Pix4D - 22min just for calibration

The `opencalibration` pipeline

DSM + ortho preview (in progress)

- Thumbnails from input images are used to quickly generate a low-resolution orthomosaic
- Also generate a DSM to go with it

What’s next for `opencalibration`

- Quality metrics - need ground truth datasets
- Shifting clusters to avoid discontinuities
  - Weight edges which were ‘cut’ in previous iterations higher
- Inverse camera model experiments
  - Right now the camera models are defined 3D → 2D
  - Makes directly optimizing camera parameters on mesh slow/complicated
  - Idea: define camera model the other way, fit, minimize, fit back
- Making actually useful outputs:
  - Surface models: 3D mesh / Digital Surface Map
  - Orthomosaic
  - Idea: a scaled up version of the thumbnail preview
- Sharing it with the team at OpenDroneMap
  - Still a long way to go before getting here

Questions

https://github.com/jkflying/opencalibration/