Photogrammetry 3D scanning on Linux

Workflow

Postup vypadá zhruba takhle:

- Nafocení dostatečnýho množství kvalitních fotek
- Vytvoření sparse point cloudu (= množina záchytnejch bodů, který jsou vyfocený na 3 a víc fotkách)
- Vytvoření dense point cloudu (= množina nezáchytnejch bodů, který jsou odvozený ze záchytnejch)
- Vytvoření meshe (3d modelu) z dense point cloudu (tohle už umí i Meshlab)

Jak fotit:

- Každej detail by měl bejt alespoň na 3 fotkách z různejch úhlů
- Fotit ze stejné vzdálenosti s 30-60% překryvem
- Vypnout automatický otáčení fotek (je lepší mít všechny fotky ve stejnym rozlišení)
- Vypnout automatickou expozici, korekci bílý, atd…
- Používat co nejmenší clonu (aby byla co největší hloubka ostrosti)
- Fotit s co největším rozlišením a co nejostřeji
- Ideálně fotit pod mrakem, když je světlo, ale nejsou žádný ostrý stíny
- Lesklý a jednobarevný plochy je potřeba zmatnit a pokrejt vzorama (křídový sprej, gafa, barevný tečky, laserový tečky,…)
- Vyhnout se odleskům
- Pokud jsou na pozadí viditelný feature, tak se scanovanej objekt nesmí vůči pozadí hejbat
- Zapnout ukládání GPS souřadnic do EXIFu (pokud to jde)

Software

- MVE + Meshlab
- Colmap + Meshlab (funguje i na Windows)
- VisualSFM + Meshlab
- OpenMVG?
- bundler+pmvs2?
- e-foto??
- Agisoft PhotoScan (placeny SW, funguje na Linuxu)
- 3df zephyr FREE (jen Windows)

Postupy

ImageMagick

- `mogrify -resize 500000@\> *.jpg` omezit rozliseni fotek na 0.5 Mpx

Meshlab

Meshlab má spoustu zajímavejch funkcí. Tady jsou vypsaný jen ty úplně nejzákladnější potřebný k sestavení meshe z pointcloudu, rychlý očištění a export:

- Filters → Remeshing → Screen Poisson Surface Reconstruction (prej nema moc smysl depth >15, osobne sem pouzival cca 11-12)
- Filters → Smoothing, Fairing, Deformation → Laplacian Smooth
- Filters → Remeshing → Simplification: Quadratic Edge Collapse Decimation (100k faces = 5MB soubor, 200k = 10MB, atd...)
- Filters → Normals, Curvatures & Orientation → Transform: Translate, Center, Set origin → Set new origin: Trackball center
- Filters→point set→estimate radius from density
    - Filters→selection→conditional vertex selection→(rad > 0.007)
    - Delete selected vertices

MVE

```
set -x
renice -n 10 $$
time (
makescene -i ./img ./scn #-m 500000
sfmrecon ./scn  #--video-matching=10
dmrecon -s2 ./scn
scene2pset -F2 ./scn ./scn/pset-L2.ply
#optional:
fssrecon ./scn/pset-L2.ply ./scn/surface-L2.ply
meshclean -t10 ./scn/surface-L2.ply ./scn/surface-L2-clean.ply
)
```

Colmap CLI bez NVIDIA + pmvs2

```
mkdir img out
nice colmap automatic_reconstructor --image_path img/ --workspace_path out/
#--use_gpu 0
colmap image_undistorter --image_path img --input_path out/sparse/0 --
output_path out/dense --output_type PMVS
pmvs2 out/dense/pmvs/ option-all
cd out/dense/pmvs/models
meshlab option-all.ply
```

OpenMVG

```
#Sequential pro po sobe jdouci fotky z videa, Global pro nenavazujici fotky
#SfM_SequentialPipeline.py ./img ./out
```

```
SfM_GlobalPipeline.py ./img ./out
```

OSM-Bundler

```
python2 ./RunBundler.py --photos=./img
python2 ./RunPMVS.py --bundlerOutputPath=./output
#python2 ./RunCMVS.py --bundlerOutputPath=./output --ClusterToCompute=10
#pro velky datasety
```

OpenDroneMap

```
docker run -it --rm -v $(pwd)/code/images:/code/images
opendronemap/opendronemap --mesh-size 100000 --force-ccd 1  --help
```