Advanced Topics in 3D Computer Vision – SS21, TUM

Challenge3

Patrick Ruhkamp p.ruhkamp@tum.de

Technische Universität München

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

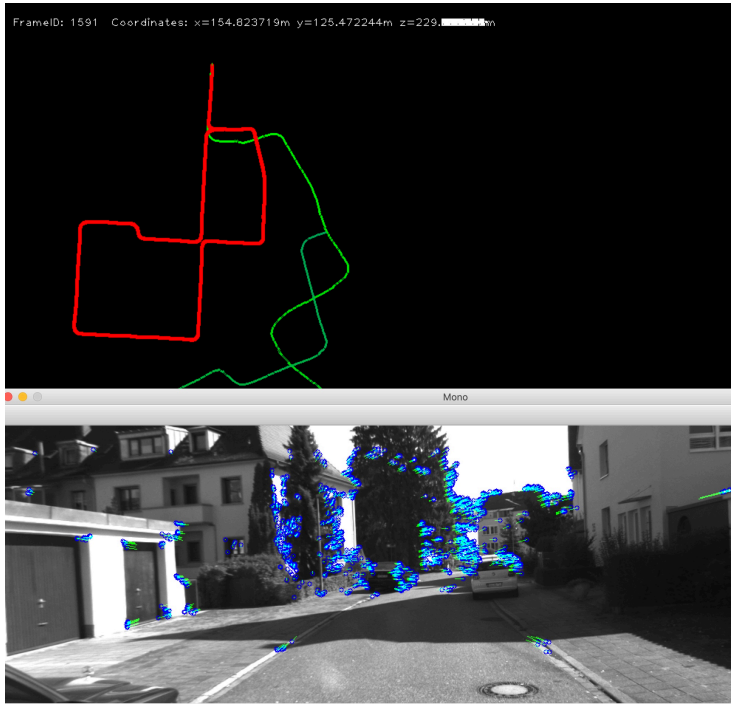
Simple Mono VO

- Task 1: Create Feature-Based VO (SIFT and ORB) [15pts]
- Task 2: Extend to Optical Flow Feature Tracking [5pts]
- Task 3: SuperPoint FrontEnd [5pts]
- Task 4: Simple Pose Graph Optimization [5pts] (OPTIONAL!)

- Deadline: 03.05.2021. 23:59 CET (Munich Time)
- http://campar.in.tum.de/Chair/TeachingSoSe21AT3DCV
- Upload:
- https://www.dropbox.com/request/DRCPRZ6tL2K0jSYbFQr9
- Data: https://syncandshare.lrz.de/getlink/fiCQBBX5hitTKyDRwAPuh5gP/Challenge3Data

Task 1:

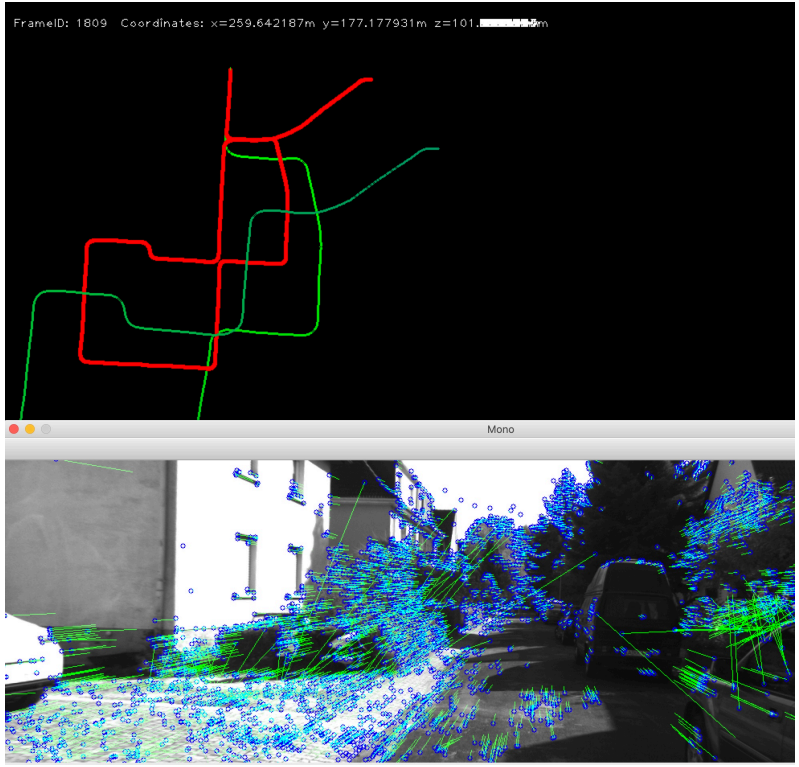
- Download Kitti Dataset
  - Poses: https://s3.eu-central-1.amazonaws.com/avg-kitti/data_odometry_poses.zip
  - Images: https://s3.eu-central-1.amazonaws.com/avg-kitti/data_odometry_gray.zip
    - Image archive weighs about 22GB (Please let me know if you have issues with the dataset size)
  - ***OR***: https://syncandshare.lrz.de/getlink/fiCQBBX5hitTKyDRwAPuh5gP/Challenge3Data
- Create Virtual Environment (https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/)
  - `python3 -m pip install --user --upgrade pip`
  - `python3 -m venv at3dcv_vo`
  - `source at3dcv_vo/bin/activate`
- `pip install numpy==1.19.4 opencv-python==3.4.2.16 opencv-contrib-python==3.4.2.16`
- Upload:
  - Python Code (runable in virtual environment as described above)
  - Screenshots from Trajectory
  - Answers to following questions (.txt file):
    1. What is the effect of using the essential matrix for pose estimation on the trajectory? And why does it work quite well for the Sequence from the Kitti Dataset? (2-5 sentences)
    2. What would be another alternative to compute the poses after the initialization? (2-3 sentences)


FrameID: 1591 Coordinates: x=154.823719m y=125.472244m z=229.
Mono

Task 2

- Extend the current Pipeline by feature tracking with optical flow

- Upload:
  - Python Code (runable in virtual environment as described before)
  - Screenshots from Trajectory
  - Answers to following question (.txt file):
    1. What do you think are the benefits or drawbacks of using optical flow to track feature points? (1-3 sentences)


FrameID: 1809 Coordinates: x=259.642187m y=177.177931m z=101.
Mono

Task 3

- SuperPoint Keypoints
  - https://github.com/magicleap/SuperPointPretrainedNetwork
  - Check the `demo_superpoint.py` script

```
print('==> Loading pre-trained network.')
# This class runs the SuperPoint network and processes its outputs.
fe = SuperPointFrontend(weights_path=opt.weights_path,
                        nms_dist=opt.nms_dist,
                        conf_thresh=opt.conf_thresh,
                        nn_thresh=opt.nn_thresh,
                        cuda=opt.cuda)
print('==> Successfully loaded pre-trained network.')
```

```
pts, desc, heatmap = fe.run(img)
```

```
def nn_match_two_way(self, desc1, desc2, nn_thresh):
  """
  Performs two-way nearest neighbor matching of two sets of descriptors, such
  that the NN match from descriptor A->B must equal the NN match from B->A.

  Inputs:
    desc1 - NxM numpy matrix of N corresponding M-dimensional descriptors.
    desc2 - NxM numpy matrix of N corresponding M-dimensional descriptors.
    nn_thresh - Optional descriptor distance below which is a good match.

  Returns:
    matches - 3xL numpy array, of L matches, where L <= N and each column i is
              a match of two descriptors, d_i in image 1 and d_j' in image 2:
              [d_i index, d_j' index, match_score]^T
  """
```

- Upload:
  - Python Code (runable in virtual environment as described before)
  - Screenshots from Trajectory

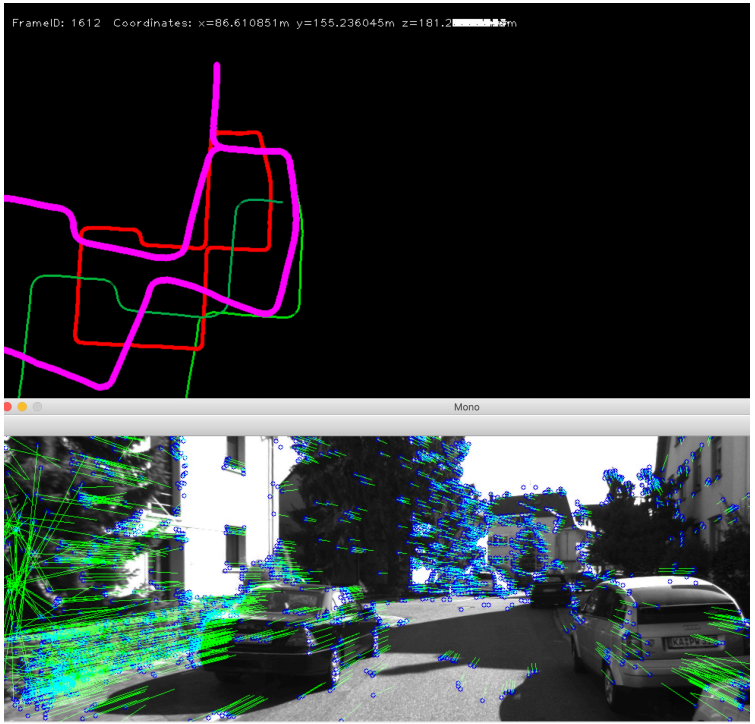
Task 4 (OPTIONAL!)

- Install Eigen3
  - `apt-get install eigen3-dev`
- Install Sophus
  - `git clone https://github.com/strasdat/Sophus.git`
  - `cd Sophus`
  - `mkdir build && cd build`
  - `cmake ..`
  - `make -j8`
- Install miniSAM (https://minisam.readthedocs.io/install_python.html)
  - `git clone --recurse-submodules https://github.com/dongjing3309/minisam.git`
  - `mkdir build && cd build`
  - `cmake ..`
  - `make`
  - `make install`

  - `cmake .. -DPYTHON_EXECUTABLE=~/at3dcv_vo/bin/python -DMINISAM_BUILD_PYTHON_PACKAGE=ON -DMINISAM_BUILD_SHARED_LIB=ON`
  - `make python_package`
  - `make install`

Task 4 (OPTIONAL!)

- Upload:
  - Python Code (runable in virtual environment as described before)
  - Screenshots from Trajectory
  - Answers to following question (.txt file):
    1. Do you think the Pose Graph Optimization improved the trajectory? Discuss how and why you come to this conclusion. (2-5 sentences)
    2. How could the trajectory be further improved? Also consider runtime in your argument? (2-5 sentences)


FrameID: 1612 Coordinates: x=86.610851m y=155.236045m z=181.2
Mono