Machine Learning Internship at Oculo

Inertial Odometry via IMU-only state estimation

John-Paul Marletta

Sup. Robert Kesten - Technical Lead

MSci Physics, Royal Holloway, University of London

Technical Presentation
October 16, 2023

Presentation Structure

1. Project Overview
2. Literature Review
3. Data formatting for TLIO
4. DevOps / Cloud Computing
5. Training TLIO
6. Results and Analysis
7. References

Project Overview

- **Position**: ML Engineer (Intern)
- **Project Title**: ‘Inertial Odometry and trajectory estimation’
- **Goals**
  - Data formatting, data pipelining, dataset for compatibility with chosen model
  - To train and test a CNN [Chen et al], , for IMU-only state estimation
  - This is to improve the overall tracking accuracy of Oculo’s product by outputting trajectory estimation when the SLAM algorithm fails.

Expectation: completely independent:

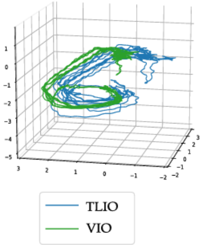
- Conducted my own LR
- Proposed papers with most promising models.

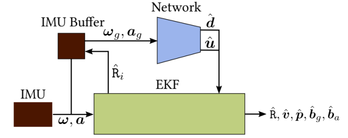
Promising model: IONet

- Approaches prior to IONet:
  - **SINS**: direct IMU data integration $\rightarrow$ **error prone**,
  - **PDR**: integration $+$ step detection algorithms
- IONet **break continuous integration**, instead segment inertial data into semi-independent windows

Literature Review

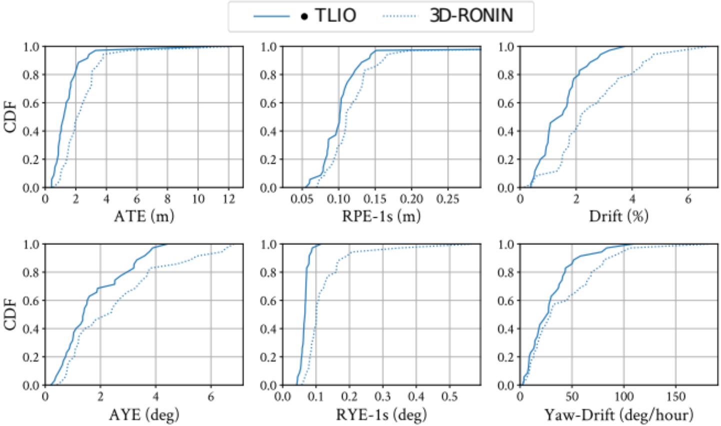
Chosen model: TLIO - Tight Learned Inertial Odometry [Liu et al.]





- UPenn & **Meta** labs - 2020
- Builds on IONet, now regressing 3D trajectories
- Propose complete state-estimation system combining **CNN/ResNet**, IMU **Buffer**, with **EKF**
- **EKF** estimates position, orientation, velocity, and IMU biases with only pedestrian IMU data
- **Reduces average yaw and position drift by 27% and 33%**, compared to best RoNIN 2019 model, using velocity concatenation approach.

Chosen model: TLIO's performance



Data formatting for TLIO

Overview and skills learnt

- **`Numpy, SciPy, pandas, matplotlib`**
- Developed **File I/O** skills : iterated over large dataset
- Approx. **700 line** script
- Greatly improved **de-bugging**, also via Run and Debug and learnt **Unit-Testing.**
- **Data pipeline**: OxIOD $\rightarrow$ **Parsing** csv files $\rightarrow$ interpolating/formatting $\rightarrow$ dicts $\rightarrow$ converting to **`pandas`** data frames $\rightarrow$ csv files $\rightarrow$ hdf5 generation script $\rightarrow$ hdf5 files

- Wrote function for ground-truth (GT) **velocity estimations** from motion capture positions and timestamps.
- **Linear interpolation** (`scipy.interpolate.interp1d`) of motion capture data aligning with IMU-only data timestamps.
- **Spherical linear interpolation of quaternions(`pyquaternion.Quaternion.slerp`)** representing rotations - more computationally efficient.

Data Formatting for TLIO

Example of Interpolating Quaternions

```python
def _interpolate_ox_vicon_quaternions(ox_imu_upsampled_data, quaternions, clipped_ox_vicon_data):
    """ - Interpolating vicon quaternions using speherical linear interpolation
        - creating dictonary: ox_interpolated_vicon_data and adding components from interpolated_quaternions
          list into separate lists inside ox_interpolated_vicon_data
    """
    ox_interpolated_vicon_data = {'timestamps': ox_imu_upsampled_data['timestamps'],
                                  'rotation_w': [], 'rotation_x': [], 'rotation_y': [], 'rotation_z': [],
                                  'translation_x': [], 'translation_y': [],'translation_z': []}


    vicon_clipped_timestamps = clipped_ox_vicon_data['timestamps']

    for timestamp in ox_imu_upsampled_data['timestamps']:
        original_index = bisect.bisect(vicon_clipped_timestamps, timestamp) - 1

        if original_index >= len(vicon_clipped_timestamps) - 1:
            original_index = len(vicon_clipped_timestamps) - 2


        if original_index < 0:
            original_index = 0


        fraction = ((timestamp - vicon_clipped_timestamps[original_index]) /
                    (vicon_clipped_timestamps[original_index + 1] - vicon_clipped_timestamps[original_index]))

        slerped_quat = Quaternion.slerp(quaternions[original_index], quaternions[original_index + 1], fraction)

        ox_interpolated_vicon_data['rotation_w'].append(slerped_quat.w)
        ox_interpolated_vicon_data['rotation_x'].append(slerped_quat.x)
        ox_interpolated_vicon_data['rotation_y'].append(slerped_quat.y)
        ox_interpolated_vicon_data['rotation_z'].append(slerped_quat.z)

    print("quaternions interpolated")

    return ox_interpolated_vicon_data
```

DevOps / Cloud Computing

Docker

- Lightweight for running TLIO remotely
- Building docker containers, images, mounting volumes
- Implemented **'keepalive'** feature in **ssh config file** to solve connection issues while copying TLIO to ec2 as **.tar.gz**
- **Adjusted container memory**, fixed issue
- Used docker container with **`conda`** to solve local system issue

AWS - EC2

- Ran docker in EC2 VM, via ssh, for CPU and memory strain
- Used **r5.2xlarge** machine: **8 vCPUs** for **multiprocessing**, 64GB
- **CLI: git bash**
- **Linux commands** including file operations and **`cat`**, **`scp`**, **`shh`**, **`ping`**, **`logs`**, **`top`**, **`ps`** etc..

Training TLIO

- **TLIO input dim:** N × 6, (N IMU samples) - gravity-aligned
- **Multiprocessing** - 7 workers
- Adam optimisation
- **80%, 10%, 10%** train, test, validate split
- batch sizes of 1024

$$L_{\mathrm{MSE}}(\boldsymbol{d}, \hat{\boldsymbol{d}}) = \frac{1}{n} \sum_{i=1}^{n} \left\| \boldsymbol{d}_i - \hat{\boldsymbol{d}}_i \right\|^2$$

- d = 3D output displacement from model, $\hat{d}$ is ground truth data, n is the number of data points in the training set

Results and Analysis

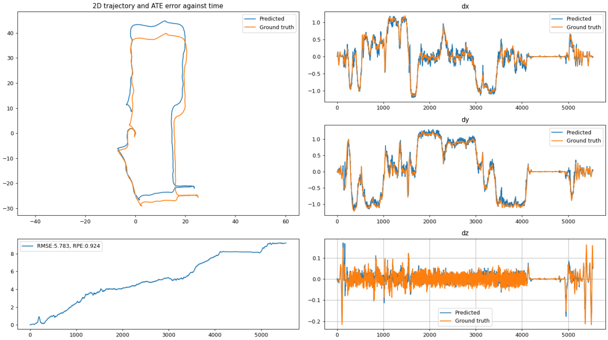


Figure: 10th training epoch: estimated vs ground truth trajectories

- **Drift**: within $0.02 - 0.05$ in m/m
- **MSE loss**: $x : (0.006 - 0.152)$, $y : (0.0004 - 0.13896)$, $z : (0.00212 - 0.17467)$
- More training was needed at this point.

Results and Analysis

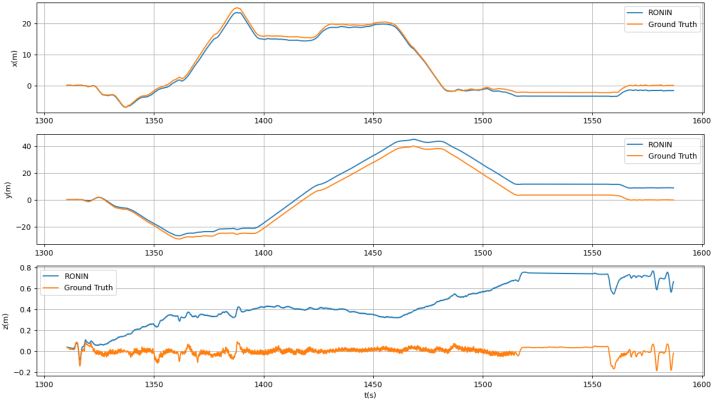


Figure: Plots of x, y, z predictions vs Ground Truth, over trajectory, at 10th training epoch

Thank you! Questions? Comments?

- Liu, Wenxin, et al. (2020)
  TLIO: Tight learned inertial odometry.
  Robotics and Automation Letters 5.4: 5653-5660.
- Chen, Changhao, et al (2018)
  IONet: Learning to cure the curse of drift in inertial odometry.
  Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1.
- Liue, Wenxin, et al. (2020)
  *h*ttps://github.com/CathIAS/TLIO