Kalman filter for height of elevation map

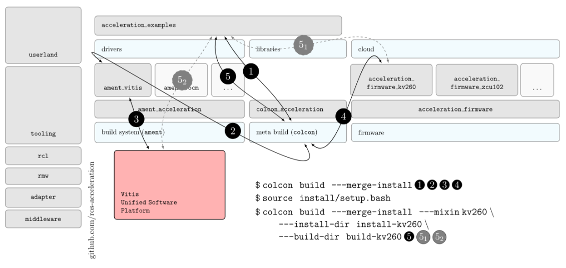
Maria Victoria Gianello

1 Introduction

In this report, an FPGA implementatiopn of a Kalman filter(KF) was designed based on the one used in [1] to estimate the height of an elevation map at a given cell. This GitHub repository was devised by the authors of the paper [2] which describes the methods used to create a 2.5D elevation map on a robot. The Kalman filter was selected because it processes large amounts of data, allowing for data parallelization, and it can also be broken down into small, parallelizable operations, which would benefit from pipelining on an FPGA to achieve low latency.

2 Design

The initial idea for the presented project was to modify the code in [3], a Robot Operating System (ROS) 2 implementation of [1]. The KF part of the algorithm responsible for height updates would have been implemented on FPGA fabric, while running the rest on the processing system (PS). This however, proved a challenge because the repositories use ROS, and to work with ROS, the Kria workflow shown in 2.1 needs to be followed. This workflow is not compatible with the Vitis flow or the Alveo U280 available board. Due to time constraints, the code could not be re-written to not use ROS1, but instead the height update of an elevation map was implemented based on the concepts discussed in the aforementioned paper and repositories.



```
$ colcon build ---merge-install ❶❷❸❹
$ source install/setup.bash
$ colcon build ---merge-install ---mixin kv260 \
    ---install-dir install-kv260 \
    ---build-dir build-kv260 ❺ 5₁ 5₂
```

Figure 2.1: Communication between ROS and HLS tools as shown in [4]

To find the section of code corresponding to KF in [3], the Elevation Mapping c++ file was examined, looking at the updatePrediction() function. This function checks if there is a new pose at the latest time, and if there is, it calls another update function in the file Robot Motion Map Updater. This function is the KF as it is taking a map, a robot pose, the noise of the robot pose, and the time as inputs. The code in this function follows the same principles as the KF described in the paper [2] where the robot pose is transformed to the map coordinate, a Jacobian is computed along with a translation variance, ...

2.1 Hardware-Software communication

The original idea was to generate a map with ROS in the PS and only implement the Kalman filter on the FPGA fabric to update the map. Since this was not possible, the Kalman filter was written in HLS and a host excecutable was created to transfer data into and out of the FPGA to mimic the original intent...

2.2 Inputs and Outputs

most of the data i found was made for ros, so i couldn't use it. To adapt the update function which is essentially the KF into HLS, the robot pose was treated as a 3x1 vector, the variance of measurements as scalars, and the map as a structure of cells. To generate the input pointcloud, a bunch of points were randomly generated, and to calculate the golden output, the points falling in a single cell were averaged. ... This meant that the input and golden output were written to files to ease the process of testing this...

2.3 Adaptation of repository function

To adapt the KF to HLS firstly, many C++ functions related to the Eigen library had to be removed since these are not compatible with HLS.

3 Results

3.1 Design Exploration

different accelerations were added to the design and their impact on accuracy, latency, and resource utilization was compared.

3.2 Accuracy

The map outputted when made up noisy pointcloud data is used in the KF was compared to a map calculated from non-noisy data. Data was created in MATLAB.

4 Lessons learned

- using the vitis flow for PS-PL interaction
- regular xrt libraries only work with kernel to kernel stream communication, while kernel to host communication has to be handled through OpenCL because the data has to pass through ...
- i would not start with a ros code if i could do things differntly

5 Conclusion

Bibliography

[1] ANYbotics. *GitHub - ANYbotics/elevation*$_m$*appingatros2*. GitHub, 2014. URL: `https://github.com/ANYbotics/elevation_mapping/tree/ros2` (visited on 10/30/2024).

[2] Peter Fankhauser, Michael Bloesch, and Marco Hutter. “Probabilistic Terrain Mapping for Mobile Robots With Uncertain Localization”. In: *IEEE Robotics and Automation Letters* 3 (Oct. 2018), pp. 3019–3026. DOI: `10.1109/lra.2018.2849506`. URL: `https://ieeexplore.ieee.org/document/8392399` (visited on 10/30/2024).

[3] Muhammad540. *GitHub - Muhammad540/elevation*$_m$*apping* : *ROS2portofRobot*$-$*centricelevationmappingforr* GitHub, 2024. URL: `https://github.com/Muhammad540/elevation_mapping/tree/main` (visited on 10/30/2024).

[4] *ROS 2-centric — KRS 1.0 documentation.* URL: `https://xilinx.github.io/KRS/sphinx/build/html/docs/features/ros2centric.html` (visited on 11/07/2024).