Project topic for C++ object oriented programming: Visual SLAM

Scope of the work: what features and functionalities will be implemented, how is the program used, and how does it work

Visual SLAM builds 3D map out of video input. The map consists out of 3D points and poses of the camera at individual frames. Juuso and Jere will implement the main SLAM workload (estimation of the map with visual odometry), Olivia will implement the visualization & UI of the estimated map, and Arundev will implement the optimization of the map with graph optimization.

The video frames will be piped into the program using the input, and the output will be visualized in a new window, supporting simple camera movement.

The high-level structure of the software: main modules, main classes (according to current understanding) and division of work and responsibilities

Visual Interface (Olivia):

The visual interface of the program consists of two parts: Map visualization and user interaction. The visualization takes up a bulk of the complexity here, but the user interaction is also important for the functionality of the product.

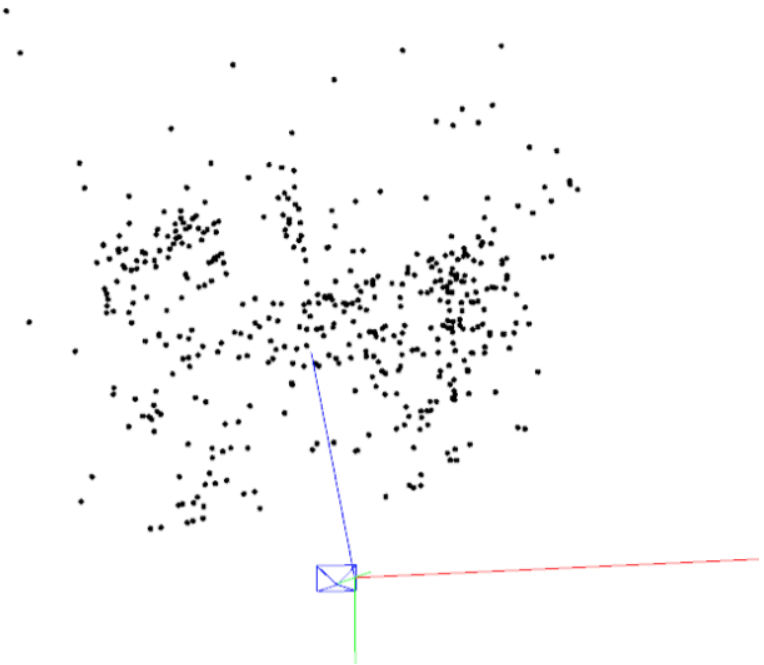
Visualization of the map (Olivia):

Visualization of the SLAM map includes visualizing the camera poses at different time steps of the video and placing the estimated points in the output in 3D space

The main classes of the visualization include:

- Viewer (Responsible for drawing the 3D output into the screen)
- Camera (Responsible for computing the visibility of points in space)

Bare-bones example of points in space



User interaction

The user interaction is very simple. As a MVP, the output window will have a number of options to control the camera position and movement. The Interface class is responsible for handling this user interaction.

Map estimation:

Map estimation consists out of two parts:

Feature extraction and matching (Juuso):
- Feature matching is done using computer vision techniques.

- Frame, Feature Extractor and Feature Matcher classes

Transform estimation (Jere):
- Transformations between camera poses are estimated using matched features and computer vision algorithms.

- Transform, Point classes

Map building (Jere & Juuso):

- Map class requires Frame, Point and Transform classes
- establishes the connections between these classes (which frames sees which points, which transforms connect which frames)

Map optimization (Arundev):

- Build optimization graph based on Map class: poses and points as nodes and transformations as edges)

- For example this could consist out of Optimizer and Graph classes (but of course, this is also up to designers choice)

‘

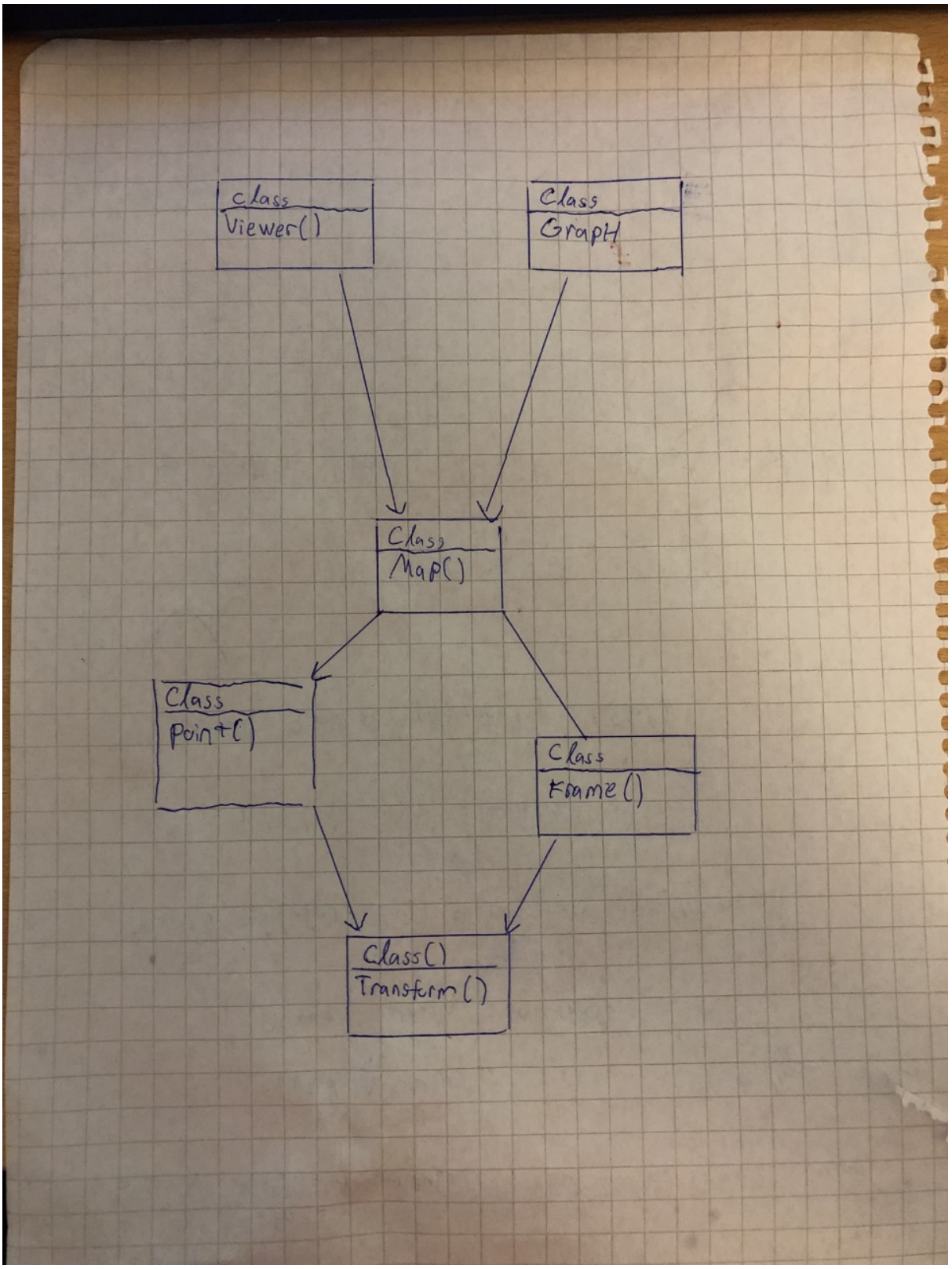


Diagram that describes a simple structure of our program.

The planned use of external libraries

We plan to use openCV for computer vision tasks, since it is a mainstream tool for it. Eigen will be used for the numerical methods and calculations. Pangolin might be used for the visualization. Also the following libraries are needed when building g2o.

- libsuitesparse-dev
- qtdeclarative5-dev
- qt5-qmake
- libqglviewer-dev-qt5

Planned schedule and milestones before the final deadline of the project:

Important Dates:

Have a meeting about the project plan

-> Project plan review with your advisor: Week 46 (starting Monday, November 14)

Olivia and Arundev learn about the visualization and optimization and test their modules with simulated data, while Juuso and Jere will implement the SLAM workload.

This week will also make sure the proper development environment is established with all of us (compiler, dependencies, cmake, merge requests).

Week 48: everyone should have their module up and running with simulated data. Each module should have their MVP ready. Putting each module together should not be difficult, but a little time will still be reserved for it later on.

Project demo to your advisor: Weeks 49 and 50 (starting Monday, December 5)

Hopefully we can get every module working together.

During the final week, we'll also build our documentation using doxygen, and touch up any existing class / function documentation to make sure everything is up to date.

Project final commit to git (deadline): Friday, December 09, 2022 at 23:59

Project peer evaluation (deadline): Friday, December 16, 2022 at 23:59