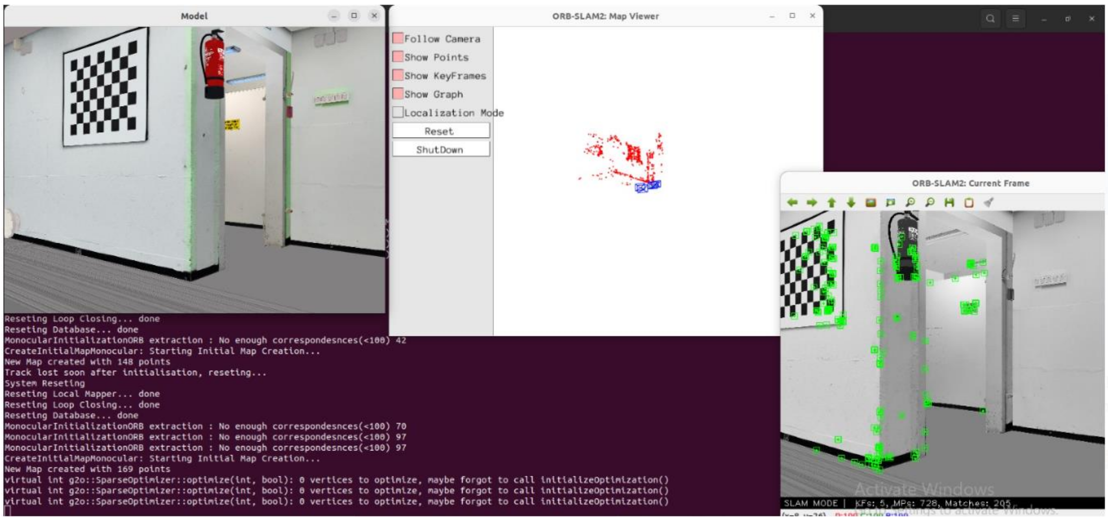
Simulator & ORB-SLAM

We have managed to successfully download the simulator, including ORB-SLAM.

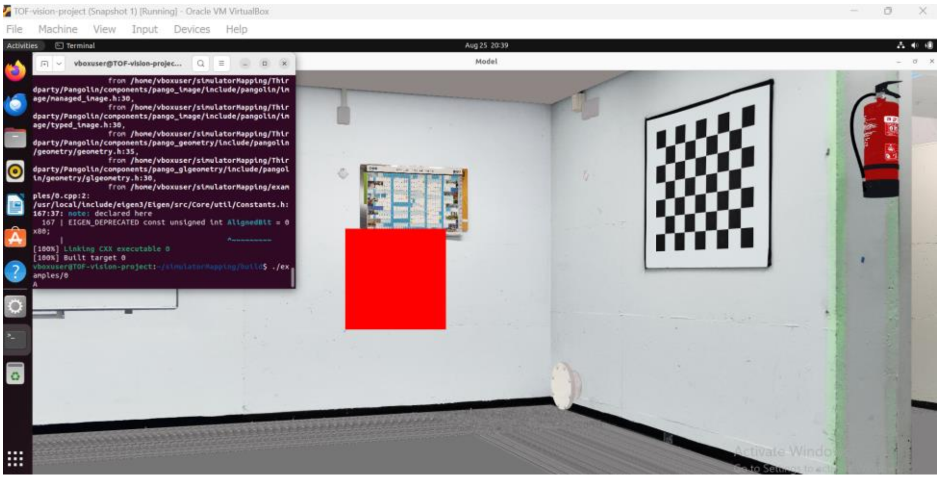
Here’s a picture from one of our initial runs:



```
Reseting Loop Closing... done
Reseting Database... done
MonocularInitializationORB extraction : No enough correspondesnces(<100) 42
CreateInitialMapMonocular: Starting Initial Map Creation...
New Map created with 148 points
Track lost soon after initialisation, reseting...
System Reseting
Reseting Local Mapper... done
Reseting Loop Closing... done
Reseting Database... done
MonocularInitializationORB extraction : No enough correspondesnces(<100) 70
MonocularInitializationORB extraction : No enough correspondesnces(<100) 97
MonocularInitializationORB extraction : No enough correspondesnces(<100) 97
CreateInitialMapMonocular: Starting Initial Map Creation...
New Map created with 169 points
virtual int g2o::SparseOptimizer::optimize(int, bool): 0 vertices to optimize, maybe forgot to call initializeOptimization()
virtual int g2o::SparseOptimizer::optimize(int, bool): 0 vertices to optimize, maybe forgot to call initializeOptimization()
virtual int g2o::SparseOptimizer::optimize(int, bool): 0 vertices to optimize, maybe forgot to call initializeOptimization()
```

```
SLAM MODE |  KFs: 5, MPs: 728, Matches: 205
```

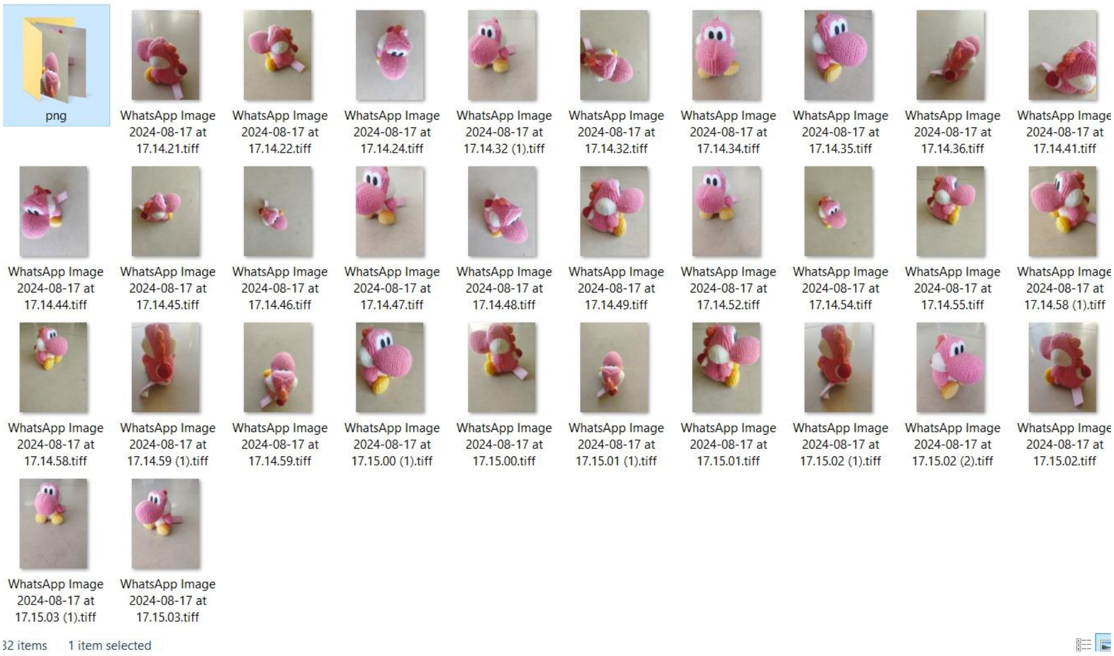
To enable us to write code for the simulator, we’ve tried several approaches. Of course, we can write code on the VM be it straight from the terminal using VIM or NANO, or downloading an IDE to the VM. However, since the VM has more limited resources and is generally slower, we’ve decided it would be better to write code from the windows host. To implement this we created a shared folder between the windows host and the ubuntu guest, we tried using symlinks and binds to automate the process of transferring files from the shared folder to the simulator dir, however, both solutions ended up breaking the “shareness” of the folder so we decided to instead use the shared folder “as is” and manually move the files. To automate this we wrote a shell script that copies (not moves) the file, and if it isn’t already in the simulatorMapping/examples/examples/CMakeLists.txt file, it adds the file, additionally it runs make in simulatorMapping/build. To test this we slightly modified the simulatorUsageTemplate.cpp code (from the windows host) by changing the size of the red square and keeping the points on the screen, after some attempts we managed to get it to work:


TOF-vision-project (Snapshot 1) [Running] - Oracle VM VirtualBox
File
Machine
View
Input
Devices
Help
Activities
Terminal
Aug 25 20:39
vboxuser@TOF-vision-projec...
Model
Activate Windo

```
                        from /home/vboxuser/simulatorMapping/Thir
dparty/Pangolin/components/pango_image/include/pangolin/im
age/managed_image.h:30,
                        from /home/vboxuser/simulatorMapping/Thir
dparty/Pangolin/components/pango_image/include/pangolin/im
age/typed_image.h:30,
                        from /home/vboxuser/simulatorMapping/Thir
dparty/Pangolin/components/pango_geometry/include/pangolin
/geometry/geometry.h:35,
                        from /home/vboxuser/simulatorMapping/Thir
dparty/Pangolin/components/pango_glgeometry/include/pangol
in/geometry/glgeometry.h:30,
                        from /home/vboxuser/simulatorMapping/exam
ples/0.cpp:2:
/usr/local/include/eigen3/Eigen/src/Core/util/Constants.h:
167:37: note: declared here
  167 | EIGEN_DEPRECATED const unsigned int AlignedBit = 0
x80;
      |                                     ^~~~~~~~~~
[100%] Linking CXX executable 0
[100%] Built target 0
vboxuser@TOF-vision-project:~/simulatorMapping/build$ ./ex
amples/0
A
```

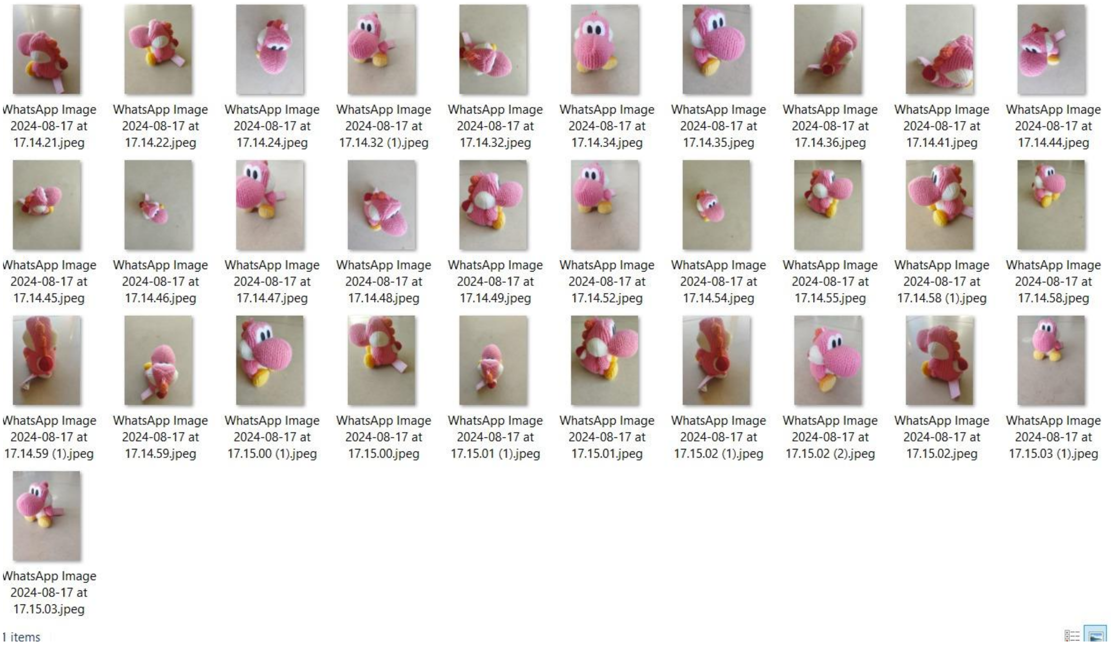
Theoretical work:

Since TOF cameras don't really use a standard image format, we decided to start of by finding a tool that would let us quickly convert image formats. The tool we settled on is "ImageMagick" – it has a relatively simple command line interface which would make it easy to use in the future, here's a usage example:



png
WhatsApp Image 2024-08-17 at 17.14.21.tiff
WhatsApp Image 2024-08-17 at 17.14.22.tiff
WhatsApp Image 2024-08-17 at 17.14.24.tiff
WhatsApp Image 2024-08-17 at 17.14.32 (1).tiff
WhatsApp Image 2024-08-17 at 17.14.32.tiff
WhatsApp Image 2024-08-17 at 17.14.34.tiff
WhatsApp Image 2024-08-17 at 17.14.35.tiff
WhatsApp Image 2024-08-17 at 17.14.36.tiff
WhatsApp Image 2024-08-17 at 17.14.41.tiff
WhatsApp Image 2024-08-17 at 17.14.44.tiff
WhatsApp Image 2024-08-17 at 17.14.45.tiff
WhatsApp Image 2024-08-17 at 17.14.46.tiff
WhatsApp Image 2024-08-17 at 17.14.47.tiff
WhatsApp Image 2024-08-17 at 17.14.48.tiff
WhatsApp Image 2024-08-17 at 17.14.49.tiff
WhatsApp Image 2024-08-17 at 17.14.52.tiff
WhatsApp Image 2024-08-17 at 17.14.54.tiff
WhatsApp Image 2024-08-17 at 17.14.55.tiff
WhatsApp Image 2024-08-17 at 17.14.58 (1).tiff
WhatsApp Image 2024-08-17 at 17.14.58.tiff
WhatsApp Image 2024-08-17 at 17.14.59 (1).tiff
WhatsApp Image 2024-08-17 at 17.14.59.tiff
WhatsApp Image 2024-08-17 at 17.15.00 (1).tiff
WhatsApp Image 2024-08-17 at 17.15.00.tiff
WhatsApp Image 2024-08-17 at 17.15.01 (1).tiff
WhatsApp Image 2024-08-17 at 17.15.01.tiff
WhatsApp Image 2024-08-17 at 17.15.02 (1).tiff
WhatsApp Image 2024-08-17 at 17.15.02 (2).tiff
WhatsApp Image 2024-08-17 at 17.15.02.tiff
WhatsApp Image 2024-08-17 at 17.15.03 (1).tiff
WhatsApp Image 2024-08-17 at 17.15.03.tiff

32 items   1 item selected



WhatsApp Image 2024-08-17 at 17.14.21.jpeg
WhatsApp Image 2024-08-17 at 17.14.22.jpeg
WhatsApp Image 2024-08-17 at 17.14.24.jpeg
WhatsApp Image 2024-08-17 at 17.14.32 (1).jpeg
WhatsApp Image 2024-08-17 at 17.14.32.jpeg
WhatsApp Image 2024-08-17 at 17.14.34.jpeg
WhatsApp Image 2024-08-17 at 17.14.35.jpeg
WhatsApp Image 2024-08-17 at 17.14.36.jpeg
WhatsApp Image 2024-08-17 at 17.14.41.jpeg
WhatsApp Image 2024-08-17 at 17.14.44.jpeg
WhatsApp Image 2024-08-17 at 17.14.45.jpeg
WhatsApp Image 2024-08-17 at 17.14.46.jpeg
WhatsApp Image 2024-08-17 at 17.14.47.jpeg
WhatsApp Image 2024-08-17 at 17.14.48.jpeg
WhatsApp Image 2024-08-17 at 17.14.49.jpeg
WhatsApp Image 2024-08-17 at 17.14.52.jpeg
WhatsApp Image 2024-08-17 at 17.14.54.jpeg
WhatsApp Image 2024-08-17 at 17.14.55.jpeg
WhatsApp Image 2024-08-17 at 17.14.58 (1).jpeg
WhatsApp Image 2024-08-17 at 17.14.58.jpeg
WhatsApp Image 2024-08-17 at 17.14.59 (1).jpeg
WhatsApp Image 2024-08-17 at 17.14.59.jpeg
WhatsApp Image 2024-08-17 at 17.15.00 (1).jpeg
WhatsApp Image 2024-08-17 at 17.15.00.jpeg
WhatsApp Image 2024-08-17 at 17.15.01 (1).jpeg
WhatsApp Image 2024-08-17 at 17.15.01.jpeg
WhatsApp Image 2024-08-17 at 17.15.02 (1).jpeg
WhatsApp Image 2024-08-17 at 17.15.02 (2).jpeg
WhatsApp Image 2024-08-17 at 17.15.02.jpeg
WhatsApp Image 2024-08-17 at 17.15.03 (1).jpeg
WhatsApp Image 2024-08-17 at 17.15.03.jpeg

1 items

We found some papers that discuss 3d-mapping using TOF input (citations at the end of this report), the papers delve into the mathematics of the matter but do not provide a tool for the task at hand. Additionally, we found a github repo dedicated to the matter (https://github.com/tum-vision/lsd_slam) but it's old (last update was 10 years ago), that would mean that there will very likely be compatibility problems. This leaves with three options:

1) Use TOF input, convert it to a format ORB-SLAM can handle (using “ImageMagick”) and use ORB-SLAM.

2) Continue searching for such projects.

3) Implement it ourselves.

Due to the time limitations of the semester, option 1 seems the most realistic. To get started with it, we needed more info about the available TOF camera as (as mentioned above) there is no single widely used format for TOF cameras and we need to know more about the cameras api.

We asked Liam about the matter, and he said that due to the time limitations, we wouldn’t use actual cameras, he advised us to instead use OpenGL to get the distance of each point and use that to emulate TOF input. This can be done but it defeats the purpose of the project. Our goal was to use TOF cameras because of their unique properties, doing it this way would mean the steps in our algorithm would be:

1) Get RGB input from a standard camera.

2) Convert this data to TOF-like data.

3) Convert this data to RGB data.

4) Use this mutilated data in ORB-SLAM.

Doing it that way would mean that our only addition to the project was losing data accuracy in a complicated manner. After telling Liam our concerns we agreed to have a zoom meeting this week regarding the future of our project.

Our work this week will greatly depend on the results of our zoom meeting with Liam, in any case, we plan to write “actual” code for the simulator – more than a simple change to an existing file. It would lay the foundation for our future work and give some experience with the api.

The papers we found:

1) 6 DoF SLAM using a ToF Camera: The challenge of a continuously growing number of landmarks Siegfried. Hochdorfer and Christian Schlegel University of Applied Sciences Ulm Department of Computer Science, Prittwitzstr. 10, 89075 Ulm, Germany email: {hochdorfer, schlegel} @hs-ulm.de (http://vigir.missouri.edu/~gdesouza/Research/Conference_CDs/IEEE_IROS_2010/data/papers/0191.pdf)

2) Robust 3D-Mapping with Time-of-Flight Cameras. Stefan May, David Dr¨oschel and Dirk Holz Fraunhofer IAIS stefan may@arcor.de. Stefan Fuchs German Aerospace Center (DLR) Inst. of Robotics and Mechatronics stefan.fuchs@dlr.de. Andreas N¨uchter Jacobs University Bremen andreas@nuechti.de (https://elib.dlr.de/62654/1/FuchsM-Iros09_3dcam.pdf)