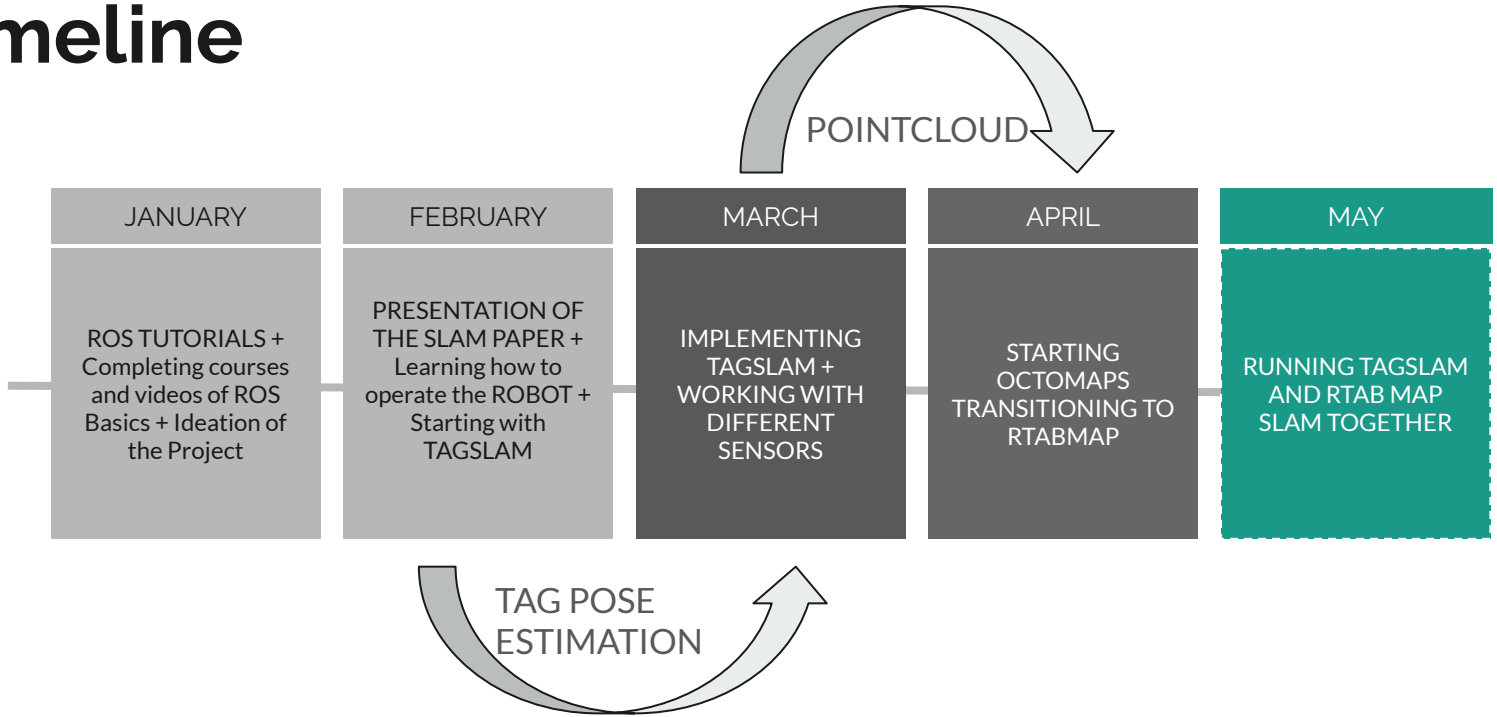
Indoor Navigation using April-Tags

Timeline


POINTCLOUD
JANUARY
FEBRUARY
MARCH
APRIL
MAY
ROS TUTORIALS + Completing courses and videos of ROS Basics + Ideation of the Project
PRESENTATION OF THE SLAM PAPER + Learning how to operate the ROBOT + Starting with TAGSLAM
IMPLEMENTING TAGSLAM + WORKING WITH DIFFERENT SENSORS
STARTING OCTOMAPS TRANSITIONING TO RTABMAP
RUNNING TAGSLAM AND RTAB MAP SLAM TOGETHER
TAG POSE ESTIMATION

ROS Tutorials + Ideation

- ROS tutorials from TU Deflt - https://ocw.tudelft.nl/courses/hello-real-world-ros-robot-operating-system/?view=lectures
- ROS tutorials from Robotics- Back end Intro: Install and Setup ROS Noetic - ROS Tutorial 1 (ROS1) (youtube.com)

- Ran multiple navigation runs on the website The Construct: Where Your Robotics Career Happens before we got access to how the robot works.

- Specifically ran their tutorial implementation of SLAM.

- Once we got access to the robot, we started working with (alpha) -3.
- Learnt how to operate the robot from Dharma Teja sir. Understood the working of IP's and normal navigation package of firebird 6.

- Started ideation of us going from the TOPOLOGICAL MAP to TOPOMETRIC MAP and then to SEMANTIC map.
- We later changed our idea to get point cloud map through RTABMAP and use the fiducial markers to navigate in the map.

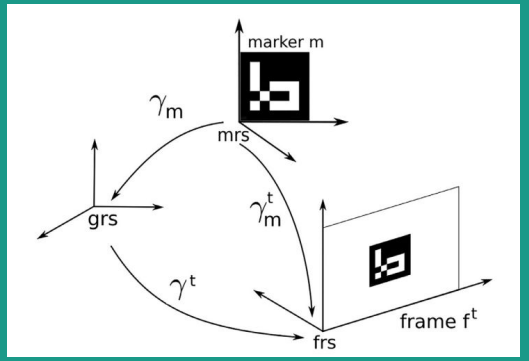
Hector SLAM

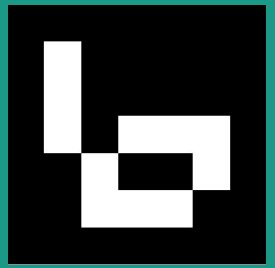
- Hector SLAM is a LIDAR based SLAM algorithm that gives a 2D occupancy grip.
- We used hokuyo 2D LIDAR and mapped out the corridor.
- This was done to get more familiar with ROS and get some experience of working with robot , NUC and properly following all the safety instructions.

SPM - SLAM

- Made a presentation on the research paper SPM-SLAM: Simultaneous localization and mapping with squared planar markers - ScienceDirect on https://docs.google.com/presentation/d/124MgWQ_jpp4yNmYi6_0Q-BUYfiI6K6Gt8YCHXgNy5uU/edit?usp=sharing .

- Learning outcomes:
  - Learnt about transformation matrices and bases.
  - Why fiducial markers give better localization and pose estimation.
  - The paper nearly solves the ambiguity problem.
  - Loop closure detection.
  - Local and Global optimizations.
  - Solving reprojection errors (difference between the actual pose of The marker and the pose that the camera sees).





TAGSLAM

- To insert tags into a map, which will be used for localization later on, we used an Open Source Repository called TagSlam.
- TagSlam did require us to first capture Tag0, and then also move the camera on the robot very slowly, while capturing the other tags, as well as several iterations, as otherwise the placement of the tags is not correct.

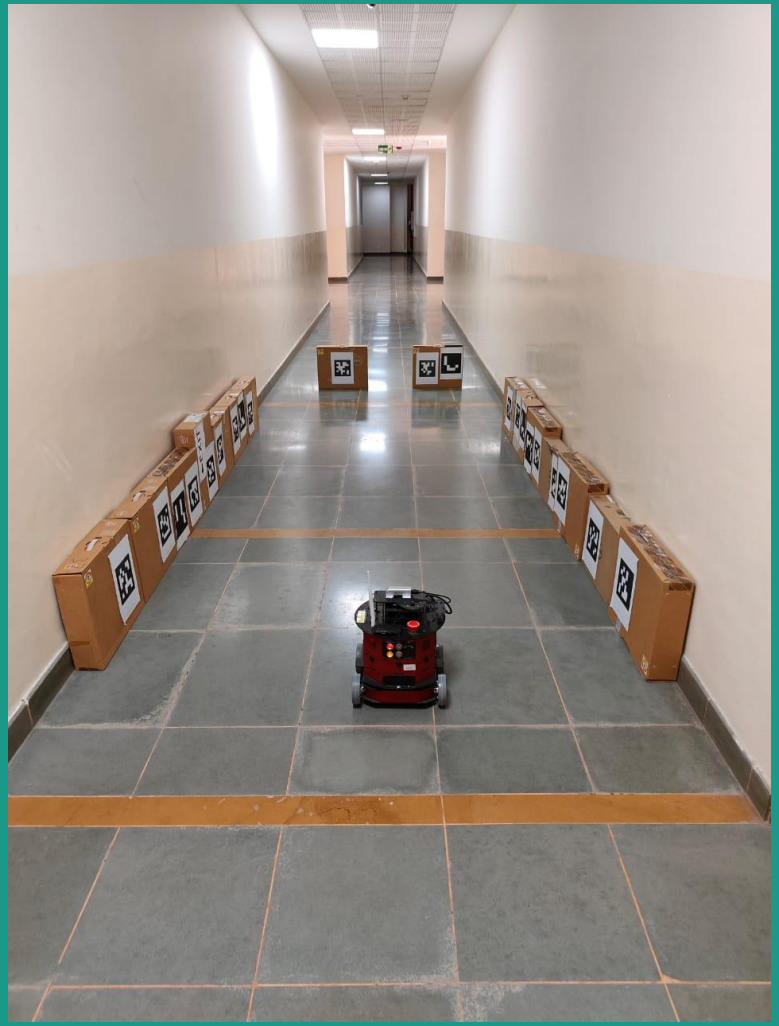
CAMERA ISSUES

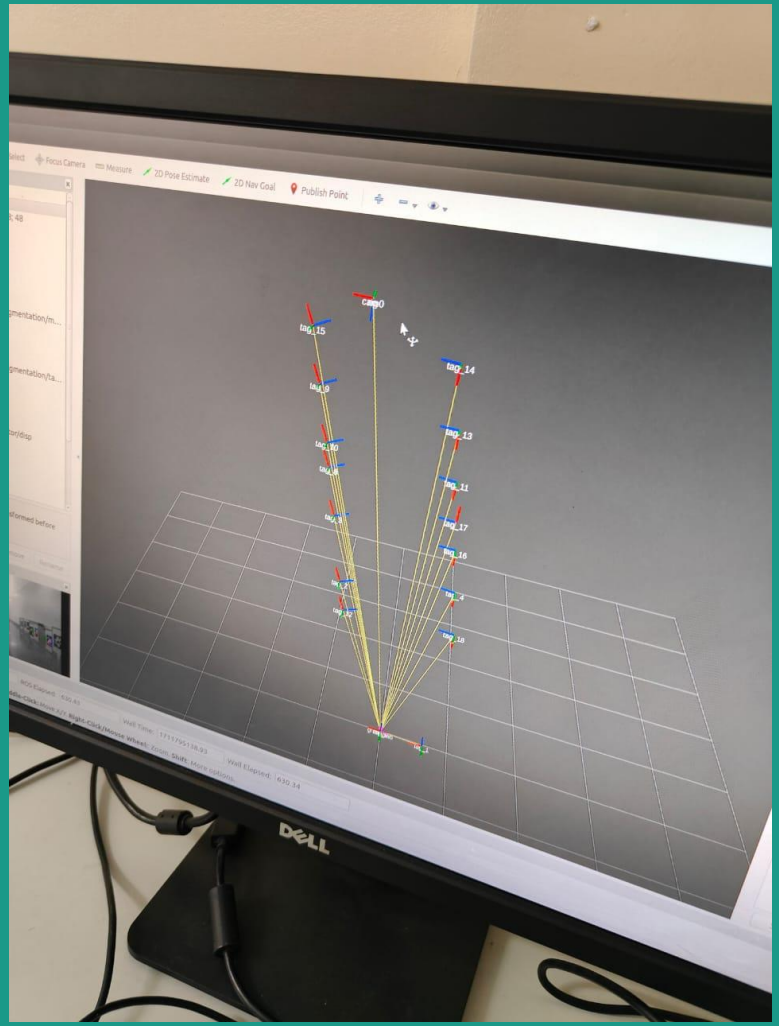
- During the implementation of TagSlam, our biggest issue was trying to get the Kinect camera working. Initially, the camera wouldn't turn on, which we found out to be a dependency issue, which could be fixed with the following command -

  sudo apt-get install cmake freeglut3-dev pkg-config build-essential libxmu-dev libxi-dev libusb-1.0-0-dev python

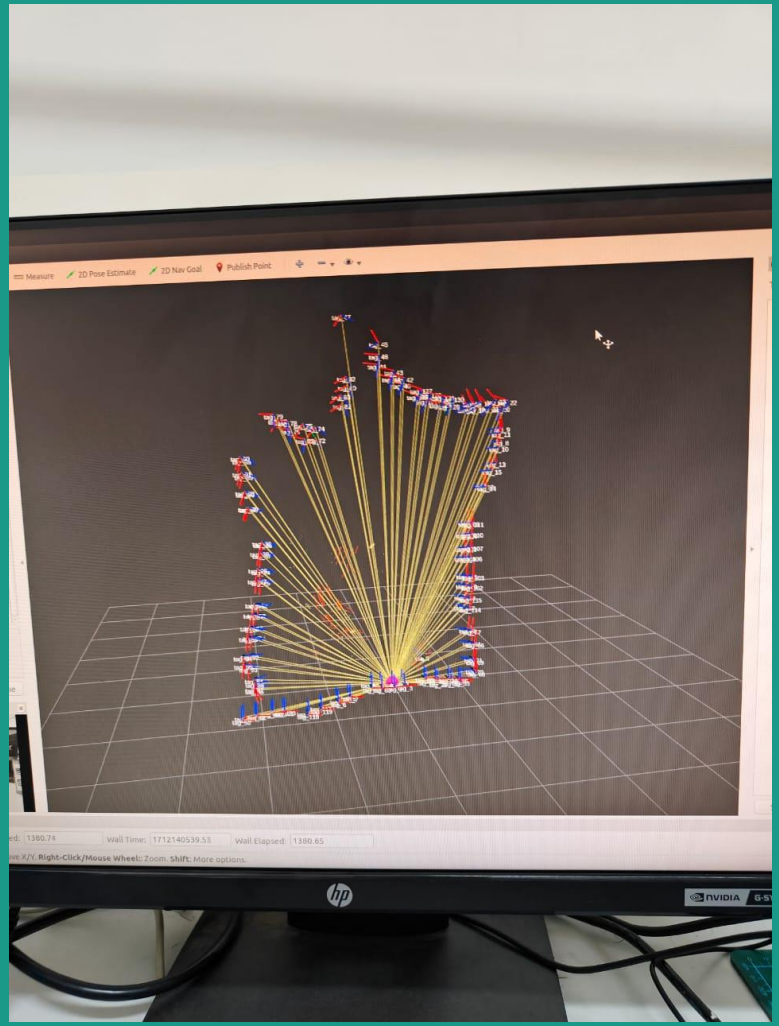
- After the Kinect was connecting, it still was not publishing data to the TagSlam node. This is because the documentation requires TagSlam to be subscribing to the camera node, which wasn't happening in our case. We eventually decided to leave this approach, and use a web camera or RealSense for all further progress.

FINAL DEMO OF TAGSLAM





TagSlam works especially well in long straight regions, with little to no rotations.



However, TagSlam does not work as well as we want when it is rotating. This is because the axis of the robot's rotation and camera don't align, as seen in the odometry data.

OCTOMAPS

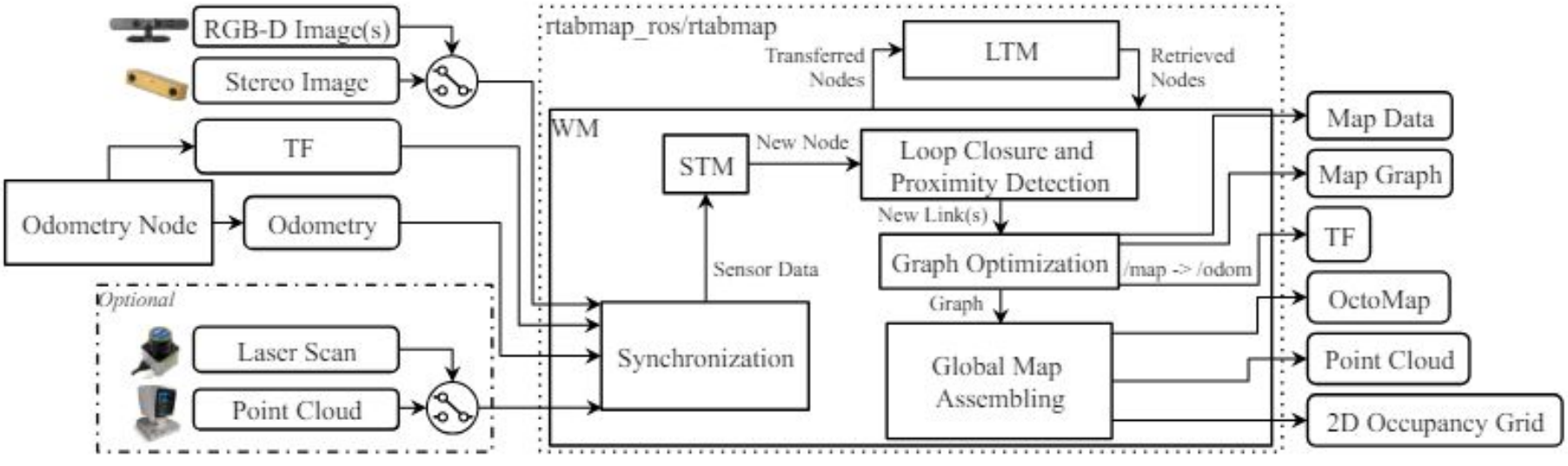
1. We initially planned on using TagSlam data directly for Octomaps.
2. We encountered some errors integrating them together.
3. We went through the source code for TagSlam as well as Octomaps, as well as trying to right a custom transform, but could not find any working solution.
4. To fix this, we proceeded to use RTabMap SLAM with TagSlam, where RTabMap is used to generate the pointcloud, and TagSlam is used for localization.

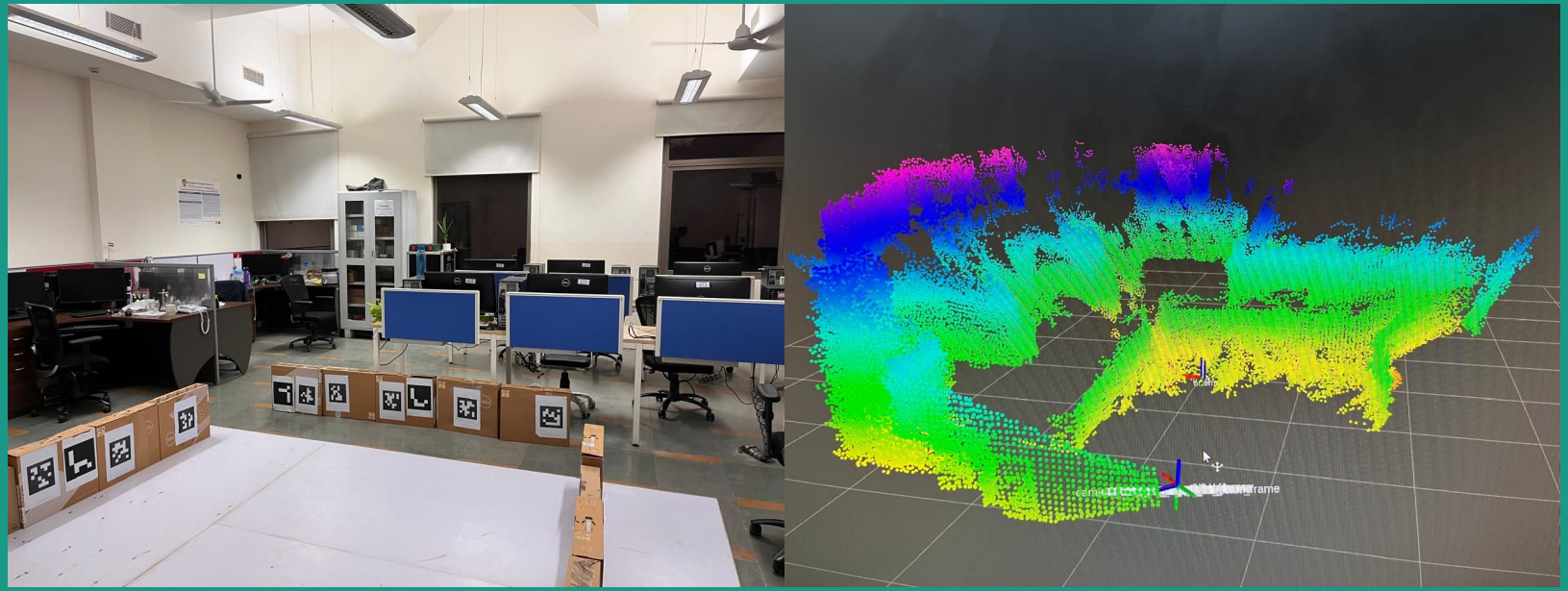
RTABMAP SLAM

- RTAB-Map (Real-Time Appearance-Based Mapping) is a RGB-D, Stereo and Lidar Graph-Based SLAM approach based on an incremental appearance-based loop closure detector.
- We used Intel Realsense D435i stereo camera.
  - Stereo image pairs
  - Depth image
  - IMU data
- Benefits of using RTAB-Map SLAM :
  - Well documented and easy to interface ROS package
  - Online processing
  - Multi-session mapping

Installation

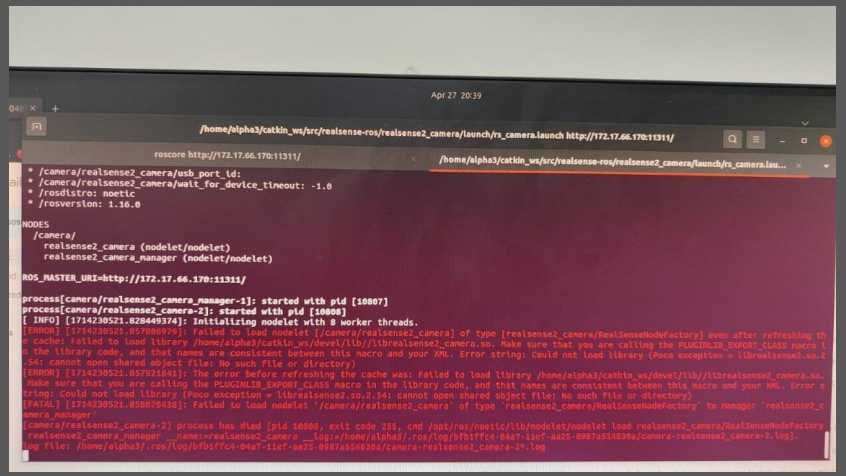
➢ sudo apt install ros-$ROS_DISTRO-rtabmap-ros


RGB-D Image(s)
Stereo Image
TF
Odometry Node
Odometry
Optional
Laser Scan
Point Cloud
rtabmap_ros/rtabmap
Transferred Nodes
LTM
Retrieved Nodes
WM
STM
New Node
Loop Closure and Proximity Detection
New Link(s)
Graph Optimization
/map -> /odom
Graph
Sensor Data
Synchronization
Global Map Assembling
Map Data
Map Graph
TF
OctoMap
Point Cloud
2D Occupancy Grid



ISSUES WITH THE REALSENSE

1. Even with TagSlam working, due to many people working on the same hardware, our data and dependencies were constantly being messed with.
2. Because of this, both RealSense nodes and Ros0x (Used to run the firebird) were deleted. Though a simple fix, we spent quite a bit of time overall fixing these issues.



COMPLETE IDEA

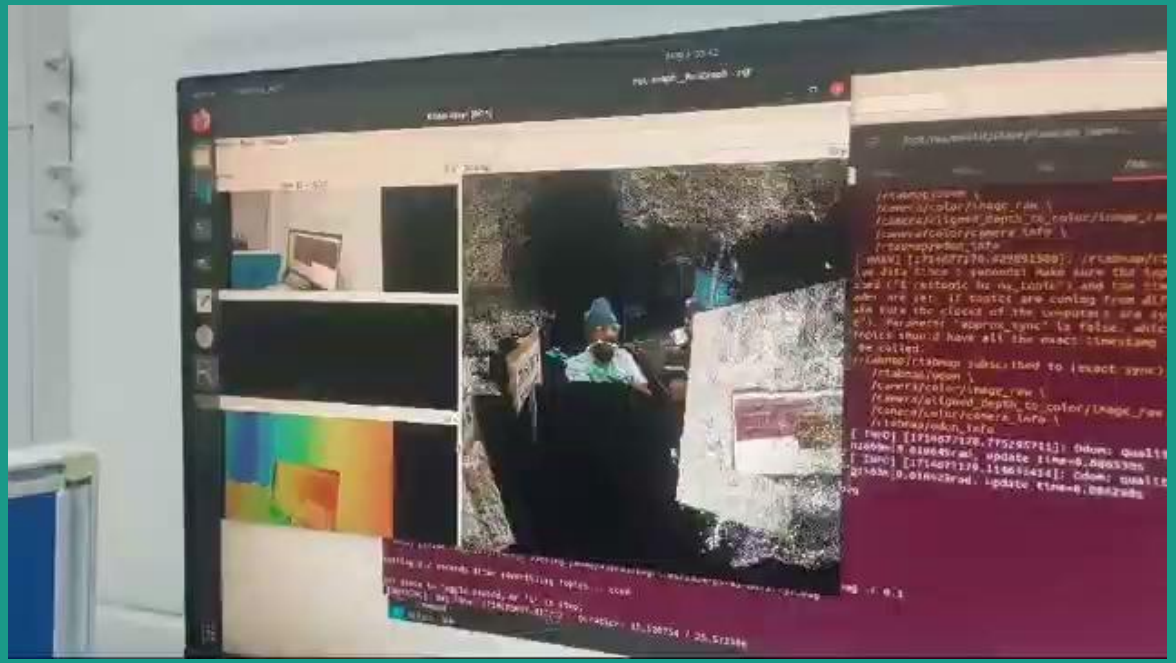
- To insert tags into the pointcloud map generated, we had to figure out how to find the tag pose relative to the point cloud map's frame.
- The map produced by tagslam and rtabmap were in different frames (their origins were translated and rotated)
- We essentially had to figure out a transform between the two maps and then convert the tag poses using the transform
- To do this, we settled on linear regression, by sampling odometry data from different positions in the environment
- The linear regressor would be trained by matching the timestamps of odometry generated by tagslam and rtabmap.

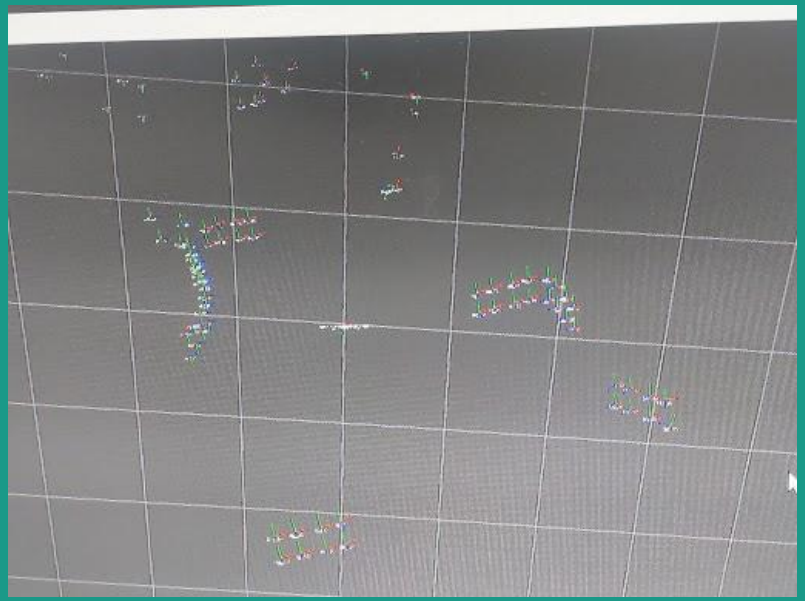
BAG FILES

A bag file contains all the data published to specific topics along with their timestamps
rosbag record <topic_names>
rosbag play <filepath>
<-r rate>

1. Our initial plan was to run both TagSLAM and RtabMap at the same time and capture data through a third node which was just a csv writer
2. However, due to hardware limitations, we couldn't run the two together without them crashing
3. To ensure that the same camera data was played to both mapping softwares, we captured the data into a bag file

Rtabmap running while a bag file of the camera data was played to it





Tagslam running on a bag file created by running the robot in this environment

Commands Used to Run Our Project

roscore

Camera Launching:

roslaunch realsense2_camera rs_camera.launch align_depth:=true unite_imu_method:="linear_interpolation" enable_gyro:=true enable_accel:=true
rosrun imu_filter_madgwick imu_filter_node _use_mag:=false _publish_tf:=false _world_frame:="enu" /imu/data_raw:=/camera/imu /imu/data:=/rtabmap/imu

Moving the Robot:

roslaunch ros0xrobot ros0xrobot_minimal.launch
roslaunch ros0xrobot ros0xrobot_teleop.launch

Recording the Bag for Camera Data:

rosbag record /camera/imu /camera/aligned_depth_to_color/image_raw /camera/color/camera_info /camera/color/image_raw /rtabmap/imu /camera/color/image_raw/compressed /tf /tf_static --repeat-latched

Commands Used to Run Our Project

TagSLAM:

```
roslaunch tagslam tagslam.launch run_online:=true
 roslaunch tagslam apriltag_detector_node.launch
 rviz -d `rospack find tagslam`/example/tagslam_example.rviz
```

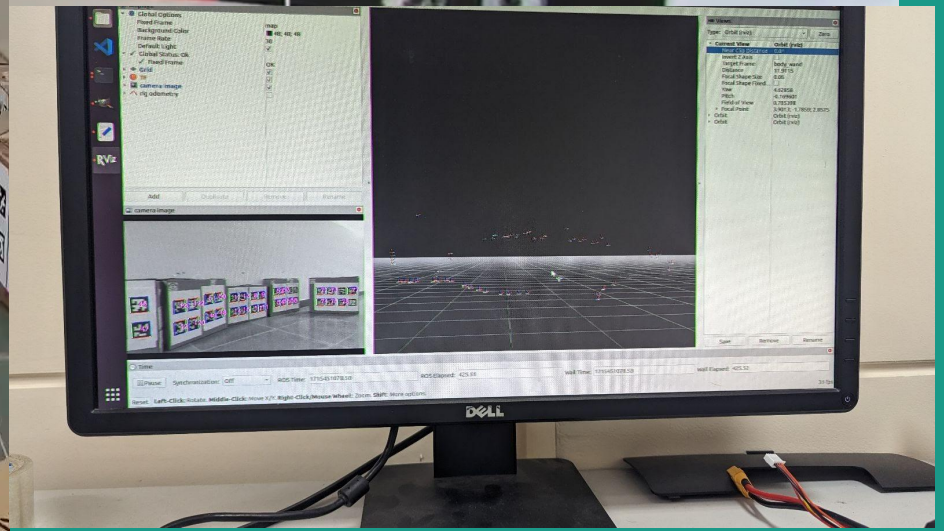
Rtabmap:

```
roslaunch rtabmap_launch rtabmap.launch    rtabmap_args:="--delete_db_on_start --Optimizer/GravitySigma 0.3"
depth_topic:=/camera/aligned_depth_to_color/image_raw    rgb_topic:=/camera/color/image_raw
camera_info_topic:=/camera/color/camera_info    approx_sync:=false    wait_imu_to_init:=false    imu_topic:=/rtabmap/imu
```

Recording Odom Data in a Bag:

```
rosbag record /rtabmap/odom
rosbag record /tagslam/odom/body_rig
```

Final Setup Photos and Videos (Environment and Maps Created)


DELL

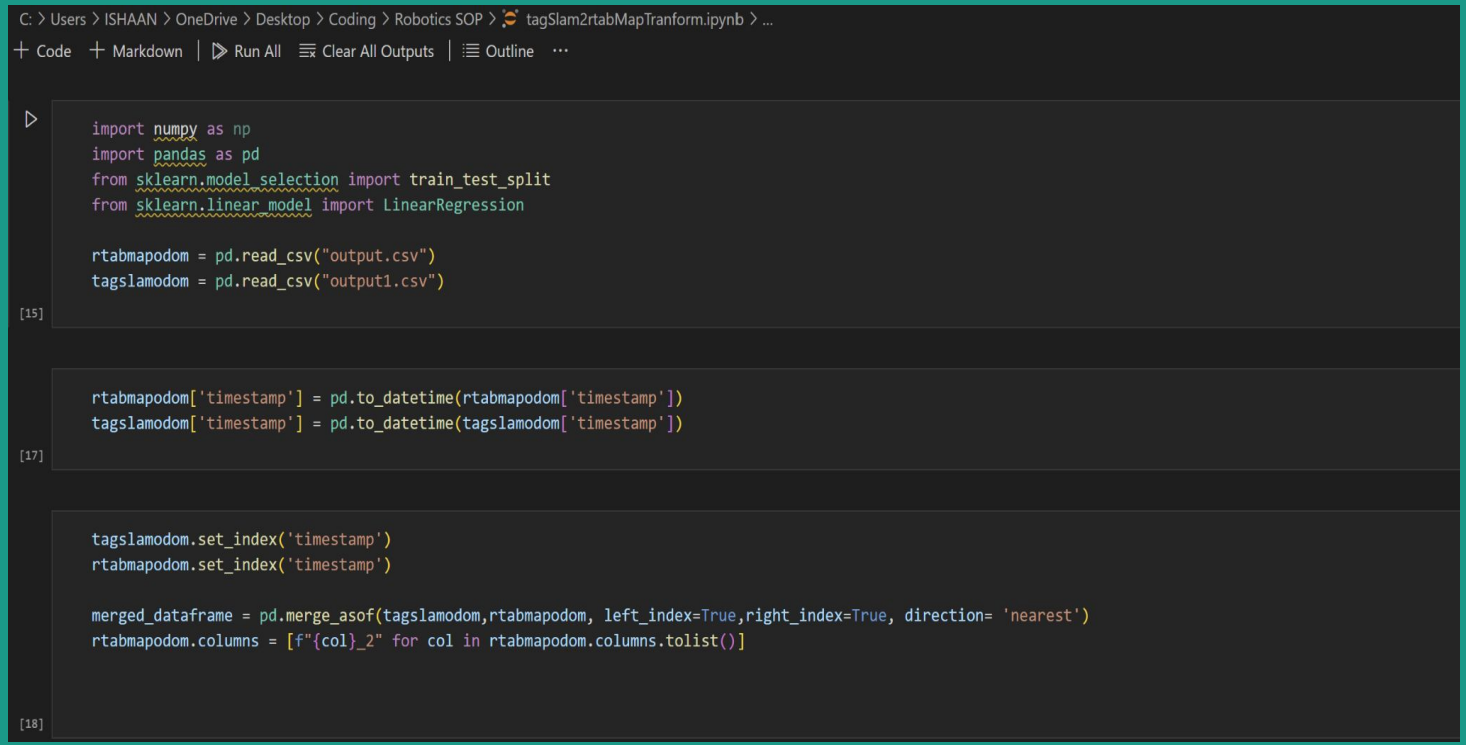
Python Script to write into CSV

rtabmap_bag_2csv.py 3

C: > Users > ISHAAN > Downloads > rtabmap_bag_2csv.py > ...

```python
import rosbag
import pandas as pd
from nav_msgs.msg import Odometry

data = {"timestamp":[], "x":[],"y":[],"z":[],"orientation_x":[],"orientation_y":[],"orientation_z":[],"orientation_w":[]}

with rosbag.Bag("tagslamodom.bag") as bag:
    print("opened bag file")
    for topic,msg, t in bag.read_messages(topics = ['/tagslam/odom/body_rig']):

            data["timestamp"].append(t.to_sec())
            data["x"].append(msg.pose.pose.position.x)
            data["y"].append(msg.pose.pose.position.y)
            data["z"].append(msg.pose.pose.position.z)

            data["orientation_x"].append(msg.pose.pose.orientation.x)
            data["orientation_y"].append(msg.pose.pose.orientation.y)
            data["orientation_z"].append(msg.pose.pose.orientation.z)
            data["orientation_w"].append(msg.pose.pose.orientation.w)
            print("wrote data")

df = pd.DataFrame(data)
df.to_csv('output1.csv', index = False)
```

Regression for Finding Transform



Sample odometry data generated

output.csv > data

```
timestamp,x,y,z,orientation_x,orientation_y,orientation_z,orientation_w
1715452080.5713675,0.03142838925123215,-0.0005561866564676166,0.0012113541597500443,7.216324776258368e-05,0.0001546695034096325
1715452080.6537626,0.03820308670401573,-0.0008370812283828855,0.0011047075968235731,-0.0001262081247227138,-6.30105153496029e-05,
1715452080.7512414,0.04301944002509117,-0.0003887545026373118,0.0006127927917987108,-0.0002769230958802516,0.0001144836021618924,
1715452080.828994,0.05027506500482559,-0.0005140303983353078,0.0014558203984051943,-0.0004151566991282693,0.000866642640961027,-0.
1715452080.939832,0.05720105022192001,-0.0009427769109606743,0.0021053715609014034,-0.0004211397995717776,0.000680536880269749,-0.
1715452081.0257308,0.06387124955654144,-0.0009454181417822838,0.0024787269067019224,-0.0003665610947214218,5.643261440186669e-05,
1715452081.1138015,0.07055614888666806,-0.0012117810547351837,0.002393348840996623,-0.000250846193369361,-0.000428730827807481,-0.0
1715452081.201668,0.07549577951431274,-0.0009127939119935036,0.0023853350430727005,-0.0004318066786536736,-0.000394098757230688,
1715452081.2947009,0.08280728757381439,-0.0012484992621466517,0.0023938785307109356,-0.0007346378964336367,0.000121893300176048,
1715452081.3870437,0.08753379434347153,-0.0009027876653635613,0.0024384239222854376,-0.0008860165328697301,0.000420476628233750,-0.
1715452081.4670393,0.09422200918197632,-0.001440616324543953,0.003297806717455387,-0.0009779621380207140,0.000407564337536693,-0.00
1715452081.54983,0.10104857385158539,-0.00175374373793602,0.0035054346080869436,-0.0009213429468661863,-0.000127784036618505,-0.0
1715452081.6287525,0.10586509108543396,-0.0016700567211955786,0.0036073436494916677,-0.0009282195681919167,-0.000494227575227526,
1715452081.7117338,0.11293832212686539,-0.0016566598787903786,0.0035636124666780233,-0.0010559005140650987,-0.000808559545889506,
1715452081.7875407,0.11754969507455826,-0.002476724795997143,0.0036375161726027727,-0.001065035437522709,-0.000549395377803805,-0.
1715452081.863674,0.12205060571432114,-0.0012006796896457672,0.0033003070857375866,-0.0011669794416049575,-0.000385206990071451,-0
1715452081.968561,0.1289563626050949,-0.0018713604658842087,0.0038618803955614567,-0.0012820732179508016,7.799199094652978e-05,-0.0
1715452082.054467,0.13606256246566772,-0.002382156439125538,0.0050161072028282793,-0.0014113813094118052,0.000126873612226783,-0.0
1715452082.1399965,0.14046937227249146,-0.0025696810334920883,0.004702906124293804,-0.0014642802573361009,-0.000262477955884504,-0
1715452082.223093,0.14768432080745697,-0.0035092609468847513,0.004254992585629225,-0.0015551013506791697,-0.000760190793833255,-0.
1715452082.3314767,0.15323232114315033,-0.0031903996132314205,0.004975878167897463,-0.0017629158903604463,-0.000499401538282498,-0
1715452082.4594743,0.16264115273952484,-0.0035342164337635044,0.006061602849513292,-0.0022682120906325834,0.000294785237595179,-0.
1715452082.55846,0.1690664291381836,-0.0063072992488741875,0.005232415162026882,-0.001874315168425819,3.563652819675036e-05,-0.001
1715452082.6423898,0.17490786314010625,-0.006669675320386866,0.0037654987536370754,-0.0015056285764163975,-0.001472259503671265,-0.0
1715452082.7188938,0.18424578011035925,-0.0028448509983718395,0.00818176381289959,-0.002148696113839632,0.000645616267415081,-0.003
1715452082.7967036,0.18804462254047394,-0.0033924616873264313,0.0058277323842048645,-0.002166921312316931,-0.000893473742997942,-0
1715452082.874227,0.19435706734657288,-0.0073763546533882625,0.0038575094658881426,-0.0017594681819672613,-0.001943409843858935,-0.
1715452082.94772,0.19926685094833374,-0.005154971498996019,0.003362041199579835,-0.0018304305442694798,-0.001746747239989315,-0.00
1715452083.019309,0.20506523549556732,-0.004583563655614853,0.006127700209617615,-0.002197722500281194,0.000209030734719432,-0.0
1715452083.0969543,0.2099345177412033,-0.004330233670771122,0.006764099001884460,-0.002154614893746626,0.000613996079415192,-0.0
1715452083.1718946,0.21675384044647217,-0.007157093845307827,0.006663164123892784,-0.001859421733552761,-0.000185022708892623,-0
1715452083.2492883,0.22179311513900757,-0.004559158347547054,0.008086083456873894,-0.001984510260246001,0.000238971138982290,-0.
1715452083.3276396,0.22454935312271118,-0.006447713356465101,0.0018019829876720905,-0.001634320492787410,-0.003460287648874508,-0
1715452083.416077,0.23271679878234863,-0.004066112916916609,0.007468858268111944,-0.002481105648388611,-0.000744339273252619,-0.0
```

Training & Accuracy Testing

```
X = merged_dataframe.drop(columns= ['x_y','y_y','z_y','orientation_x_y','orientation_y_y','orientation_z_y','orientation_w_y','timestamp_x'])
y = merged_dataframe.drop(columns= ['x_x','y_x','z_x','orientation_x_x','orientation_y_x','orientation_z_x','orientation_w_x','timestamp_x'])


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
#not required to split data after testing, just want to see error once
#in final implementation, use full dataset to train and the n predict on new points only

tranform = LinearRegression()
tranform.fit(X_train, y_train)
```

[20]

```
...
```

```
from sklearn.metrics import mean_squared_error

y_pred = tranform.predict(X_test)
error = mean_squared_error(y_test, y_pred)
print(error)
```

[23]

```
...  0.0004899099607537202
```

References and Contributions ..

SPM_SLAM -

RTABmap SLAM - https://arxiv.org/abs/2403.06341

Finding Transform by Linear Regression - https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8271986

TOPIC NAME	CONTRIBUTED BY
ROS Tutorials + ROS Basics + Robot Operations	Everyone
SPM - SLAM Paper presentation	Everyone
Practicing ROS with Hector SLAM	Dev Chheda and Shubham Gupta
Presentation of the book	Shubham Gupta and Ishaan Kale
Kinect Setup and Calibration	Gobind Singh and Shubham Gupta
USB Camera Setup and Calibration	Ishaan Kale and Gobind Singh
TagSLAM Installation	Ishaan Kale
TAGSLAM	Ishaan Kale, Gobind Singh, Dev Chheda and Shubham Gupta
Realsense Setup and Calibration	Shubham Gupta and Dev Chheda
RTABMAP installation and setup	Dev Chheda and Shubham Gupta
RTABMAP SLAM	Everyone
Trial Recording and Extracting Data from the Bag Files	Ishaan Kale, Shubham Gupta and Gobind Singh
Final Recording and Extracting Data from the Bag Files	Ishaan Kale, Shubham Gupta and Dev Chheda
Linear Regression and Node on the Data	Ishaan Kale