CHAPTER 1
INTRODUCTION

This paper aims to simulate a map of the environment that a drone traverses and optimise the trajectory of its movement. Furthermore, since the paper focuses on the utilisation of lidar-equipped drones in a forest landscape, we take a look into how lidar scanned data point map clouds are pivotal in the forestry analysis, namely, canopy cover, gaps, tree density, etc.

All the data used in this project has been obtained from the Open Topography LIDAR Dataset. The simulations have been generated on MATLAB version R2022b. A variety of predefined functions have been imported to carry out specific tasks. The functions and algorithms that are elementary and serve pedagogically fundamental content have been explained at length in the "Appendices" section.

Chapter 2 serves as an introduction to LIDAR systems

Chapter 3 deals with the implementation of SLAM algorithms and pose graph optimization and detection of the optimal trajectory of the drone and generation of the environment map.

Chapter 4 deals with implementation of forest analysis model.

Link to the simulation:
https://github.com/Gaurangee-Parashar/MATLAB-Simulation.git

CHAPTER 2

LIDAR SYSTEMS

LiDAR refers to “light detection and ranging” or “laser imaging, detection, and ranging”. It is a remote sensing method for determining ranges (variable distance) by targeting an object or a surface with a laser and measuring the time for the reflected light to return to the receiver. It can also be used to make digital 3-D representations of areas on the Earth's surface and ocean bottom of the intertidal and near coastal zone by varying the wavelength of light. The output data of LiDAR sensors, often called point cloud data, is available with 2D (x, y) or 3D (x, y, z) positional information.

It is primarily of two types. They are topographic and bathymetric LiDAR. Topographic lidar typically uses a near-infrared laser to map the land, while bathymetric lidar uses water-penetrating green light to also measure seafloor and riverbed elevations.

Components of a LiDAR system includes:

1) **Laser source and laser detector:**

The laser source generates the energy of the pulses. Near infrared wavelengths are used for most terrestrial lidar applications. Blue green wavelengths are used for bathymetry mapping as these wavelengths can penetrate water up to 40 metres depending on the water clarity. The lasers used in the lidar system are low energy and are eye-safe. High pulse repetition rates enable faster acquisition of data and/or higher point cloud density. The higher the point cloud density (typically given in points per m2), the better the achievable resolution with a LiDAR system. For instance, 4 points/m2 correlates to 0.5 m of ground sample distance. If pulses are strong enough, faster scanning can enable an aircraft to fly at higher altitude while still effectively mapping the terrain. This yields higher swath widths, which accelerates mapping throughput, thereby reducing time and flight costs. The laser detector or receiver detects the laser light pulses that are reflected back from the target objects. The scanning mechanism is designed to generate a consistent stream of laser pulses. The laser pulses are reflected off of a mirror (either rotating or scanning).

2) Timers:

The timers record the exact time the laser pulse leaves and returns to the scanner. The timing electronics have to be very precise to produce accurate data. Each pulse sent out can have up to multiple returns as it reflects off of objects on the surface. Each of the returns must be precisely timed to ensure accurate measurements for each point.

The round-trip time of flight (τ) between the transmission of an outgoing laser pulse and the arrival at the receiver of the pulse reflected by the target are used to calculate the target range (R) based on the speed of light in vacuum (c) an the average group refractive index of the optical path between the lidar system and the target (n):

$$R=(c/2n)\tau$$

Lidar systems have limited effective range because the back scattered optical signal weakens with target range, such that returns from very distant targets are too weak for the photoreceiver to detect. The effective range of a lidar system therefore depends on the sensitivity of its photoreceiver and the strength of optical signal returns as a function of target range.

3) Global Positioning System (GPS):

The Global Positioning System (GPS) records the precise X,Y,Z location of the scanner. To improve the accuracy most lidar systems use a fixed ground reference station or a Continuous Operating Reference Station (CORS). The data from the ground station or CORS has a known location and is used to correct and improve the data collected by the sensor. The GPS data is later post-processed and the precise position of the sensor at approximately every second throughout the flight can be calculated, typically with minimal error (3 to 4 cm). The GPS together with the IMU (see below) allow for the direct georeferencing of the points.

4) Inertial Measurement Unit (IMU):

The IMU contains an accelerometer, gyroscope, and magnetometer sensors that measure the velocity, orientation, and gravitational forces. The IMB constantly records the pitch, roll, and yaw of the aircraft. This data is used to determine the precise angle and location of the lidar system to ensure the distances to surfaces are correctly calculated.

5) Computer system:

A reliable computer system is required to make sure that all of the individual components of the system are working properly. The computer integrates the data from the laser system, the GPS and the IMU to produce the lidar point data. Most lidar systems include an integrated computer system along with software for the flight planning, integrating the IMG/GPS data and converting the sensor data into actual X,Y,Z coordinates.

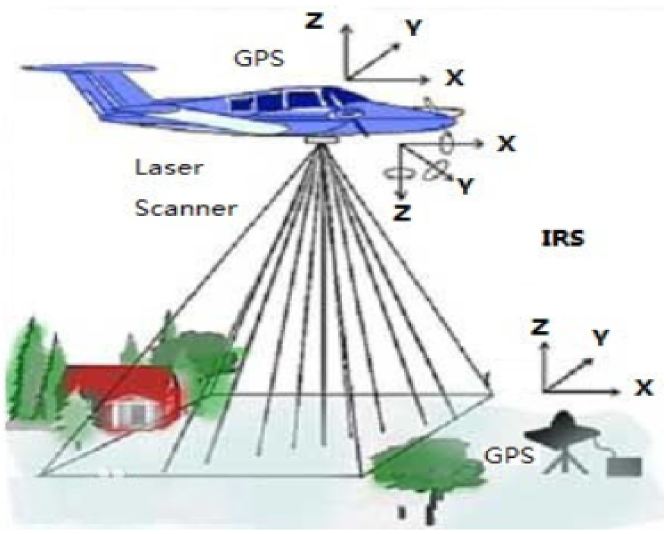


Fig 2.1: LiDAR system and its components

LiDAR systems have advanced considerably in recent years. Early commercial units were capable of operation at 10 kHz (10,000 points per second) and were large and bulky. Newer systems are more compact, lighter, and can process multiple laser returns in air, allowing for pulse rates to exceed 1 MHz. Multiple returns occur when a pulse strikes a target but is not completely reflected or absorbed, resulting in a portion of the pulse continuing to lower objects where it can also be reflected (see Figure 3). Multiple return systems can increase the amount of collected data significantly and enhance the ability to map 3D structures such as forest canopy, tree crowns, and other vegetation features.

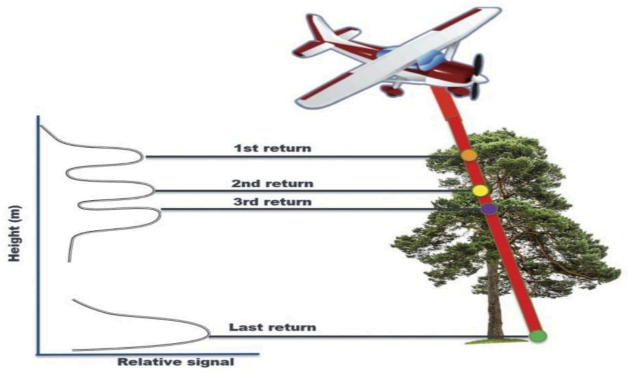


Fig 2.2: Multiple returns from a single laser pulse

CHAPTER 3

DETECTION OF THE OPTIMAL TRAJECTORY OF THE DRONE AND GENERATION OF THE ENVIRONMENT MAP

The SLAM algorithm estimates a trajectory by finding a coarse alignment between downsampled accepted scans, using fast point feature histogram (FPFH) descriptors extracted at each point, then applies the iterative closest point (ICP) algorithm to fine-tune the alignment. 3-D pose graph optimization, from Navigation Toolbox™, reduces the drift in the estimated trajectory.

3.1 - Extracting FPFH descriptors from data points

1. Compute lidar scans from the data at each waypoint of the drone using the helperCreateDataset function. The function outputs the lidar scans computed at each waypoint as an array of pointCloud objects (that is, an array of point cloud data maps). Since successive scans have high overlap, skipping a few frames can improve algorithm speed without compromising accuracy.
2. **Downsampling lidar scans** is achieved by using the box grid filter which downsamples the point cloud by averaging all points within each cell to a single point. Specify the size for individual cells of a box grid filter, in metres.

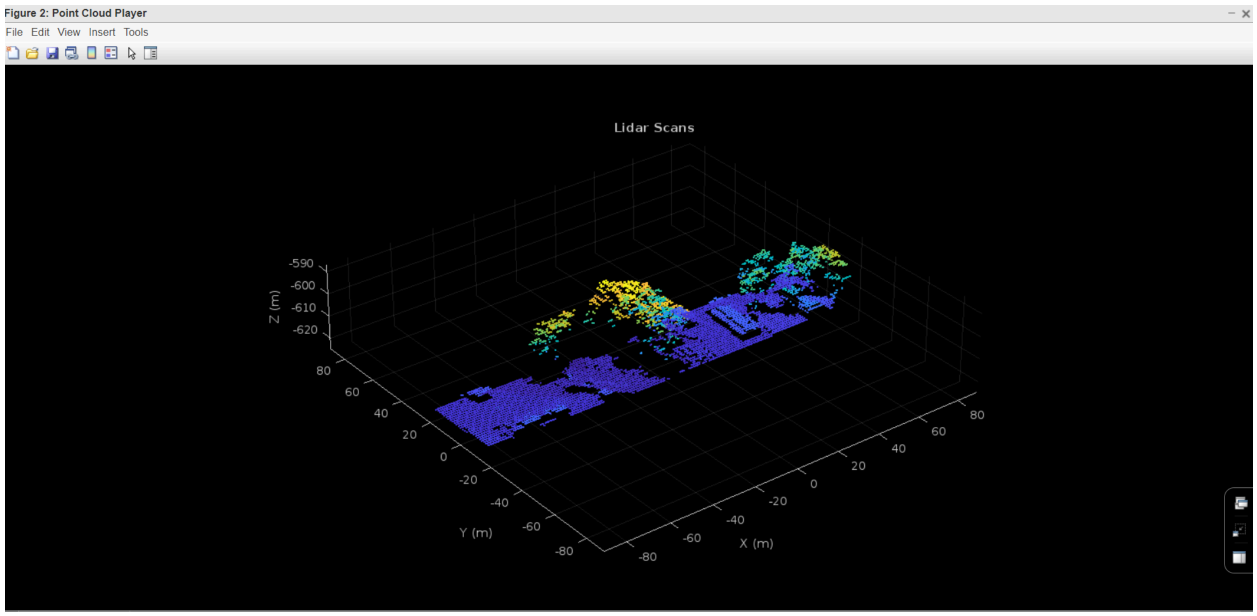


Fig 3.1: Downsampled Lidar scans

3. **Extraction of FPFH descriptors** for each valid point cloud, from a given neighbourhood size which is a user-specified parameter using the K-nearest neighbour (KNN) search method.
4. This is followed by **creation of submaps** which is achieved by aligning sequential, accepted scans with each other in groups of the specified size nScansPerSubmap, using the pcalign function. Using submaps can result in faster loop closure queries.

3.2- Loop closure using ICP

5. Then, we specify the radius around the current robot location to search for loop closure candidates. Using the helperEstimateLoopCandidates function, we can estimate matches between previous submaps and the current scan . A submap is a match, if the RMSE(root mean square error) between the current scan and submap is less than the specified value of loopClosureThreshold. The previous scans represented by all matching submaps are loop closure candidates.
6. Then,we compute the relative pose between the current scan and the loop closure candidate using the **ICP registration algorithm**. The loop closure candidate with the lowest RMSE is the best probable match for the current scan. A loop closure candidate is accepted as a valid loop closure only when the RMSE is lower than the specified threshold.

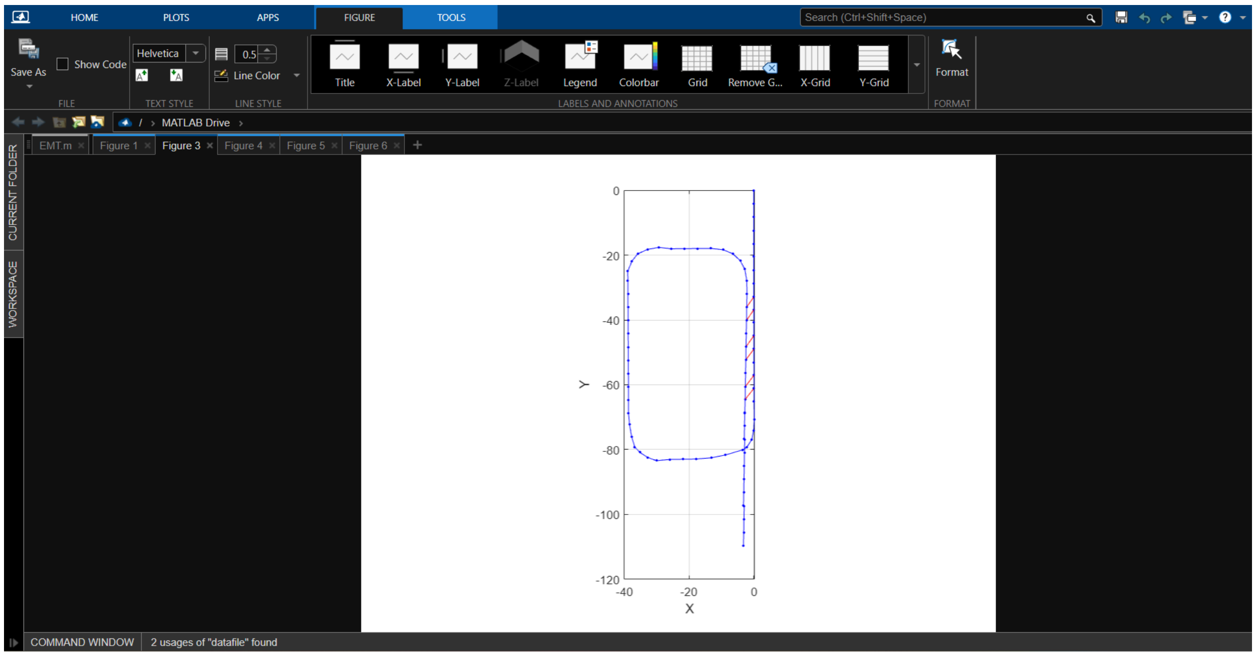


Fig 3.2: Loop closure using ICP

3.3 - Pose Graph Optimization

7. **Pose graph optimization:** After that, using a function called optimizePoseGraph (Navigation Toolbox) we reduce the drift in the estimated trajectory and find the optimal trajectory.

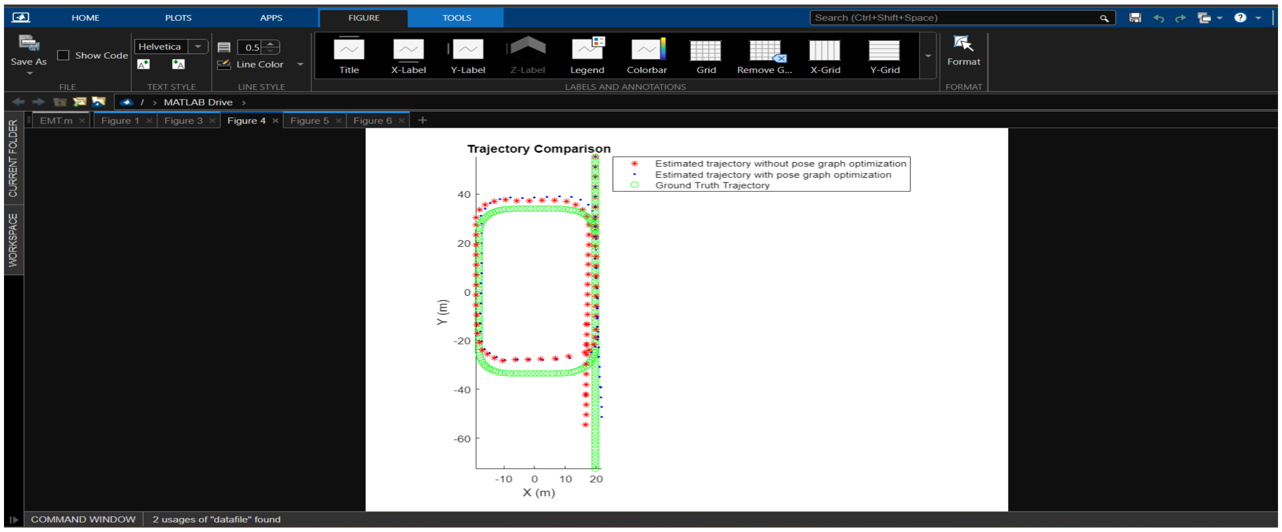


Fig 3.3: Optimisation of drone trajectory

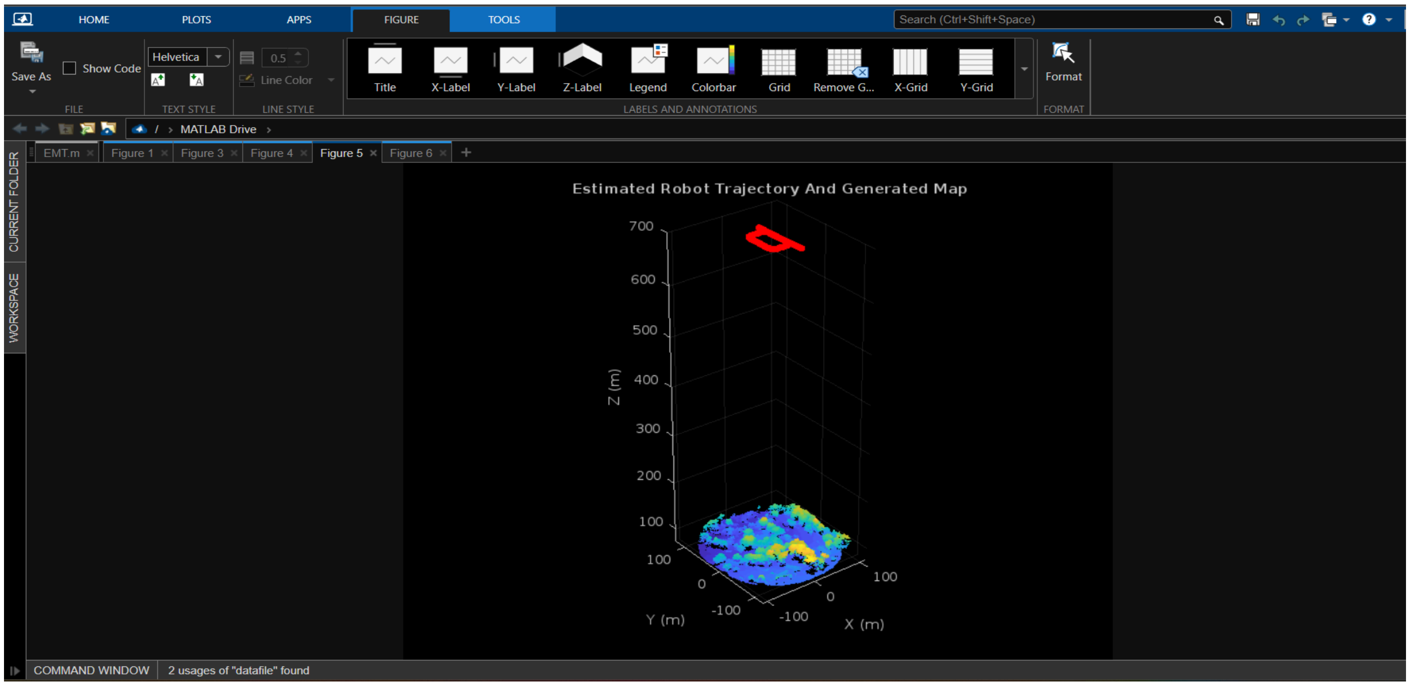


Fig 3.4: Visualisation of the estimated trajectory and generated map.

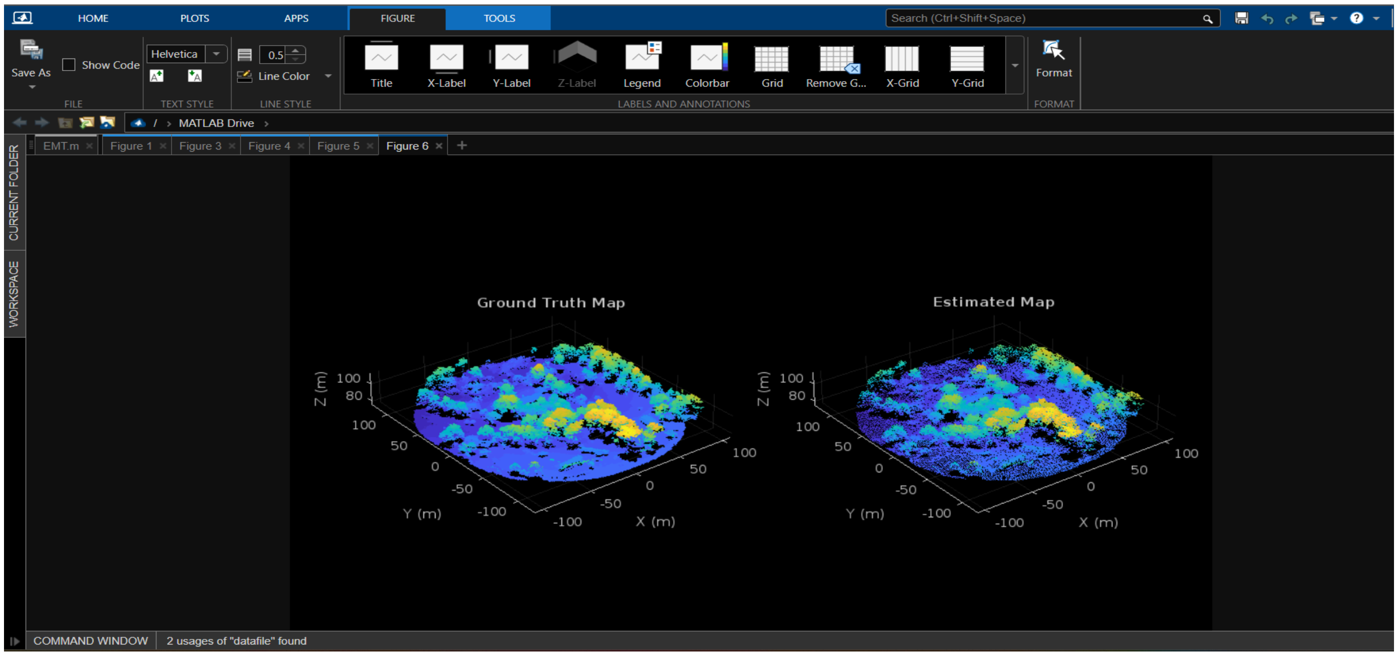


Fig 3.5: Comparison between ground truth map and estimated map

The simulation results are a clear indicator of the fact that use of ICP with pose graph optimisation yields an environment map with greater levels of accuracy.

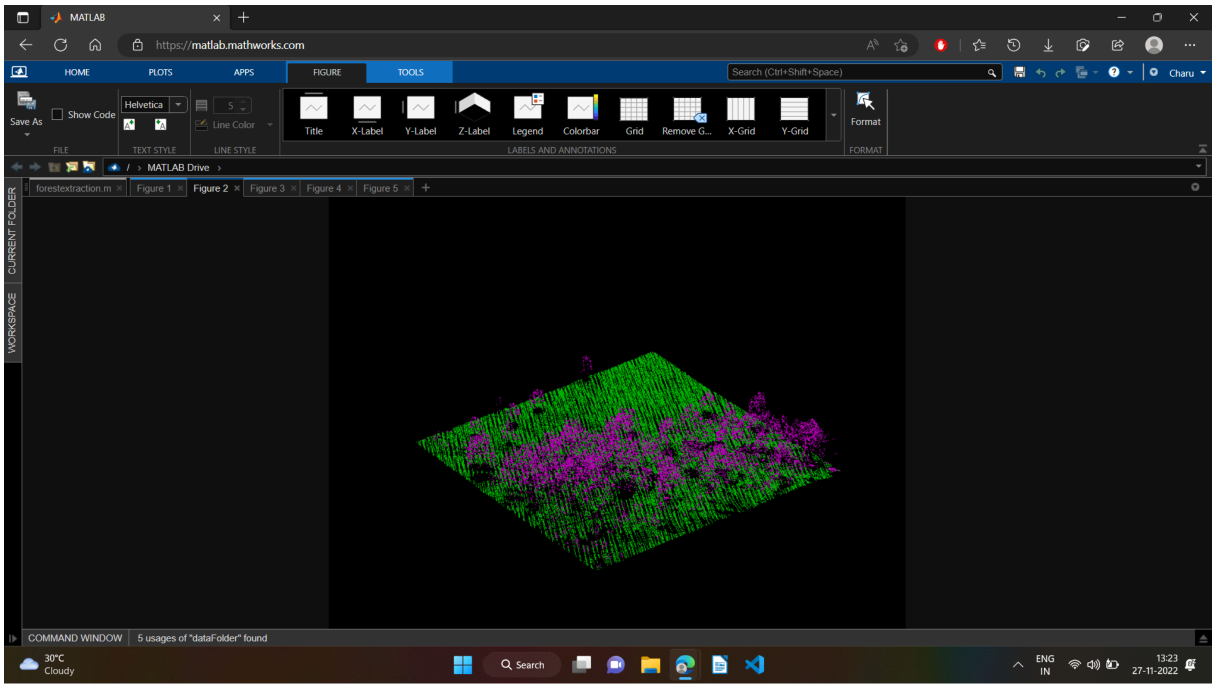
CHAPTER 4

IMPLEMENTATION OF FOREST ANALYSIS MODEL

4.1-- Data pre-processing

Prior to employing the point cloud data map of the environment as input to the forest analysis model, it has to undergo some pre-processing steps. It is essential for extraction of accurate output. These steps include:

1. Firstly, ground segmentation is performed on the input data by using a predefined function called <u>segmentGroundSMRF</u>. Ground segmentation refers to the process of separating the input data points into ground and non-ground points.



5 usages of "dataFolder" found

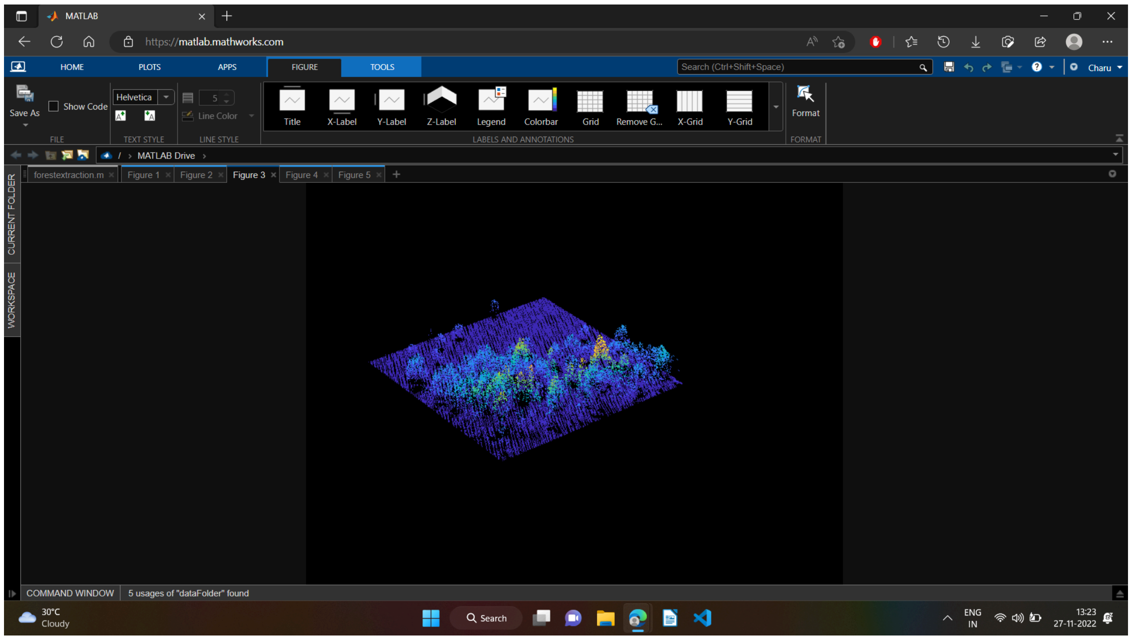
Fig 4.1: Ground segmentation

2. In order to obtain a unique set of point cloud data points, a function called <u>groupsummary</u> is used on the resultant data for the elimination of duplicate points along the x and y axes.

3. Then, considering the non-ground points as the input and creating an interpolant using the <u>scatteredInterpolant</u> object, we estimate the ground at each point. The elevation

normalisation is performed by subtracting the interpolated ground elevation from the original elevation.

(Interpolation basically refers to the process of finding a function for an existing set of data points and then using the said function to estimate values at other data points.)



5 usages of "dataFolder" found

Fig 4.2: Visualisation of environment map after duplicate point removal

4.2-- Extraction of forest metrics

On the set of pre-processed data points, a helper function called helperExtractForestMetrics is used.

(helperExtractForestMetrics function divides the input point data into grids based on the provided gridSize. Also, this function assumes that all points with a normalised height lower than cutoffHeight are ground and the remaining points are vegetation. Here, parameters like gridSize and cutoffHeight are specified by the user.)

The forest metrics computed by the function are:

- Canopy Cover (CC) — It is the proportion of the forest covered by the vertical projection of the tree crowns. Calculate it as the ratio of vegetation returns relative to the total number of returns.

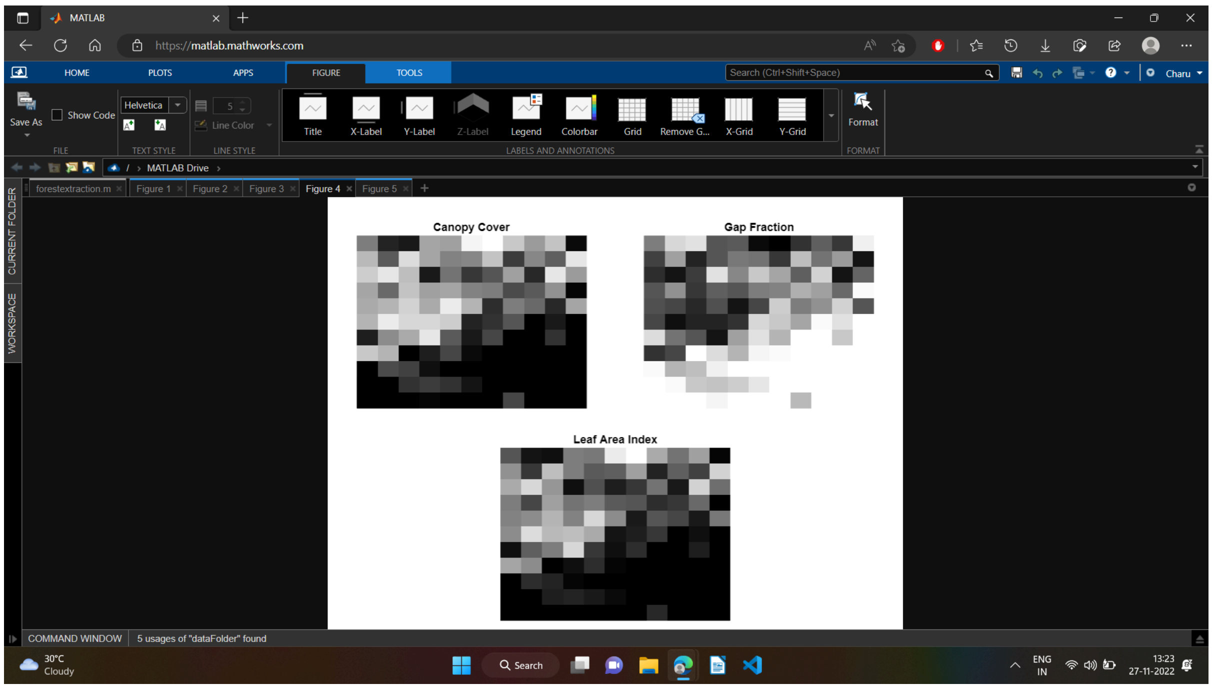
- Gap fraction (GF) — It is the probability of a ray of light passing through the canopy without encountering foliage or other plant elements. Calculate it as the ratio of ground returns relative to the total number of returns.

  (CC and GF are complementary to each other.)

- Leaf area index (LAI) — It is the amount of one-sided leaf area per unit of ground area. This metric is mostly applicable in case of broad-leaf trees. Also, it is a good indicator of the amount of biomass present in an area.

$$\mathbf{LAI=(-\cos(ang)*\ln(GF))/}\boldsymbol{k}$$

  Here, ang is the average scan angle, GF is the gap fraction, and k is the extinction coefficient, which is closely related to the leaf-angle distribution.)



5 usages of "dataFolder" found

Fig 4.3: Extraction of forest metrics from the input map

4.3- -Extraction of individual tree attributes

In order to extract various attributes regarding individual trees, firstly we need to identify and segment the individual trees present in the environment map. This can be done by following the given steps:

1. Using the pc2dem function we generate canopy height models(CHMs), which are raster representations of the height of trees, buildings, and other structures above the ground topography.

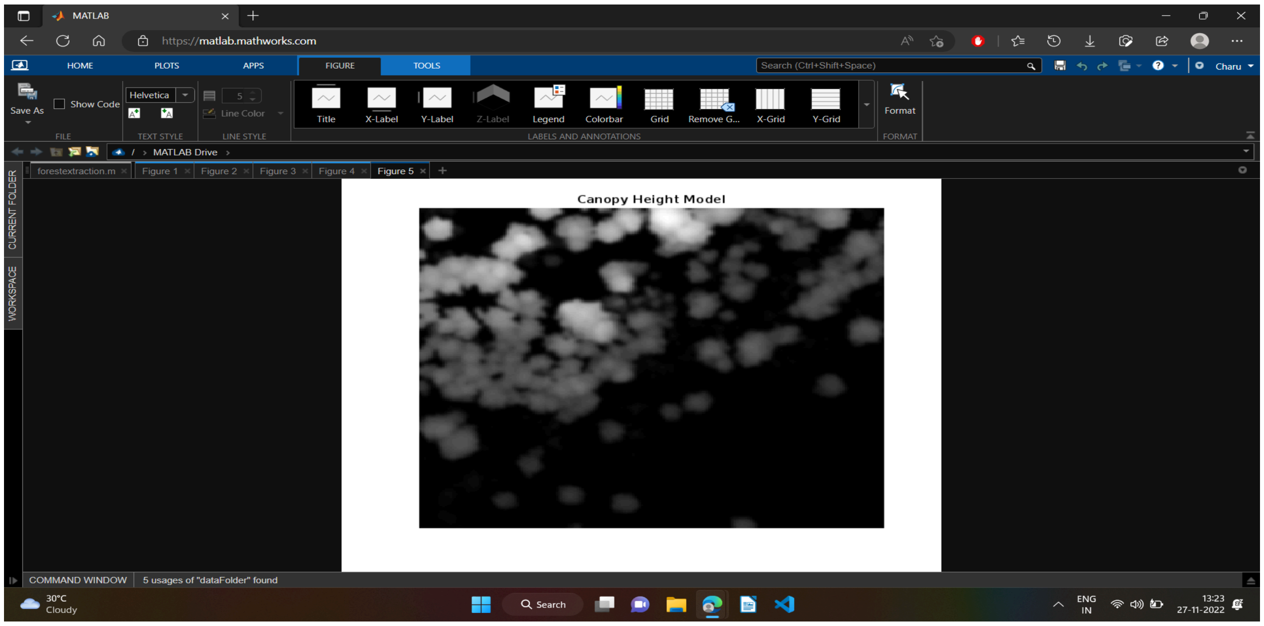


Fig 4.4 4.4: Visualisation of the generated CHM

2. Then, a helper function called helperDetectTreeTops is used on the CHM. This function helps us to detect the tree tops by finding the local maxima within variable window sizes in a CHM. For tree top detection, the helper function considers only points with a normalised height greater than minTreeHeight (a parameter which is specified by the user).

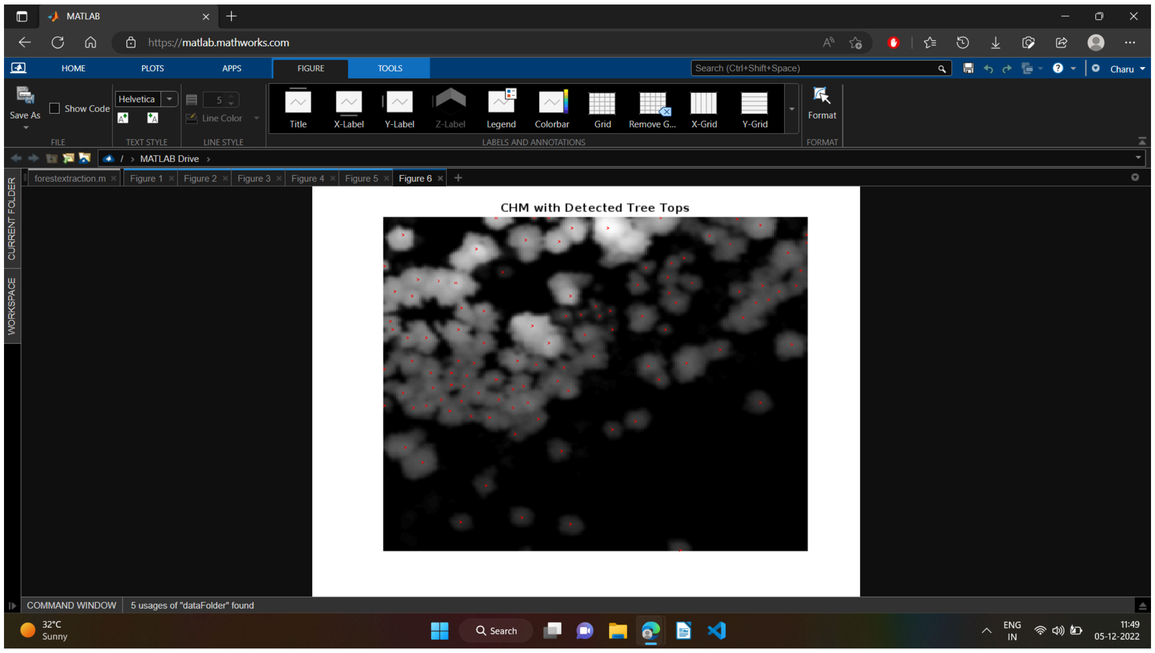


Fig 4.5: Identification of individual trees in CHM

3. This is followed by segmentation of individual trees, using the helperSegmentTrees helper function on the data. This function utilises marker-controlled watershed segmentation to segment individual trees.

   (Watershed segmentation: It refers to an image processing technique without the use of CNN. In this technique, firstly, a binary marker image with tree top locations indicated by a value of 1 is created. Then, filtering of the CHM complement by minima imposition is done to remove the minima that are not tree

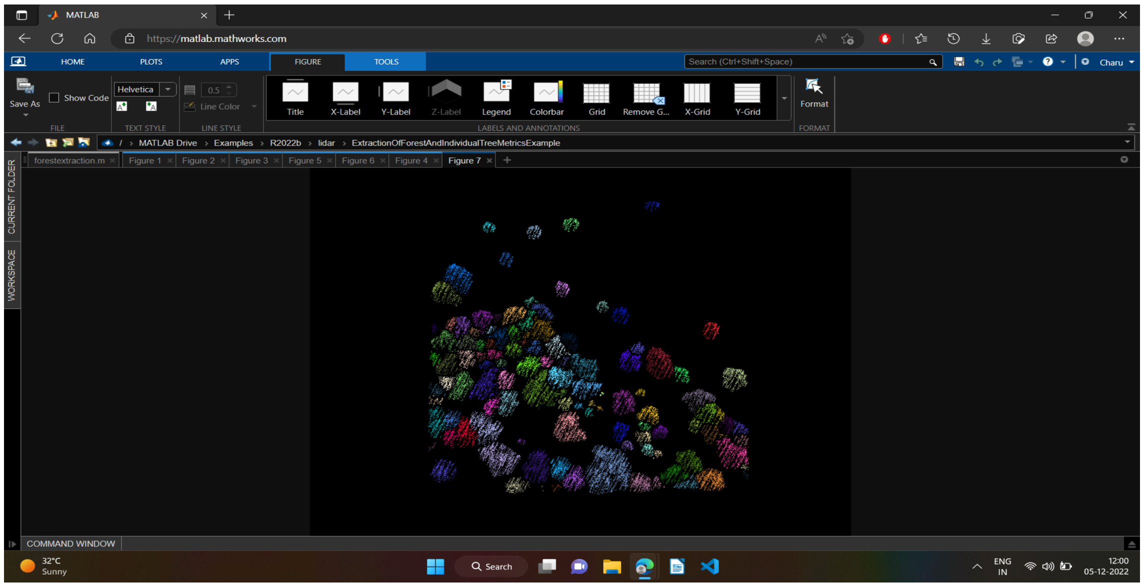
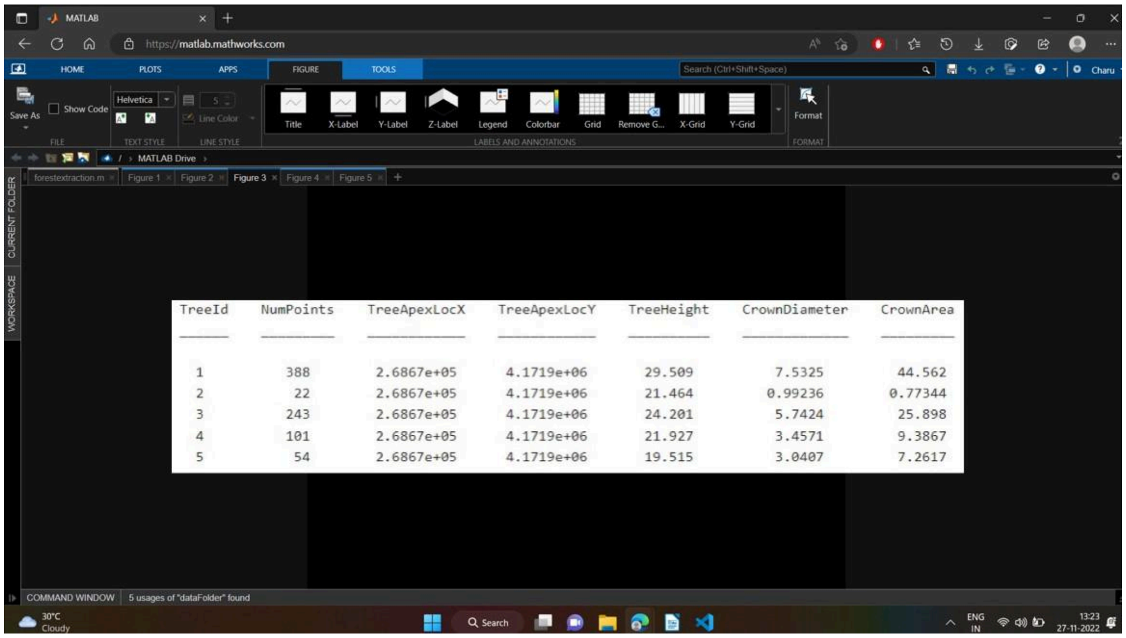

MATLAB
https://matlab.mathworks.com
HOME
PLOTS
APPS
FIGURE
TOOLS
MATLAB Drive › Examples › R2022b › lidar › ExtractionOfForestAndIndividualTreeMetricsExample
forestextraction.m
Figure 7
COMMAND WINDOW

Fig 4.6: Segmentation of individual trees

4. Then, attributes of individual trees are extracted using the helperExtractTreeMetrics helper function, attached to this example as a supporting file. First, the function identifies points belonging to individual trees from labels. Then, the function extracts tree attributes such as tree apex location along the x- and y-axes, approximate tree height, tree crown diameter, and area. The helper function returns the attributes as a table, where each row represents the attributes of an individual.


```
TreeId    NumPoints    TreeApexLocX    TreeApexLocY    TreeHeight    CrownDiameter    CrownArea
______    _________    ____________    ____________    __________    _____________    _________

  1          388        2.6867e+05      4.1719e+06       29.509          7.5325         44.562
  2           22        2.6867e+05      4.1719e+06       21.464         0.99236        0.77344
  3          243        2.6867e+05      4.1719e+06       24.201          5.7424         25.898
  4          101        2.6867e+05      4.1719e+06       21.927          3.4571         9.3867
  5           54        2.6867e+05      4.1719e+06       19.515          3.0407         7.2617
```

Fig 4.7: Extraction of individual tree attributes

APPENDIX I

ITERATIVE CLOSEST POINT (ICP) ALGORITHM

Given two sets of points P = {p1, . . . , pM} and Q = {q1, . . . , qN } in R^d, we optimise a rigid transformation on P (represented using a rotation matrix $\mathbf{R} \in R^{d\times d}$ and a translation vector $t \in R^d$) to align P with Q:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{M} (D_i(\mathbf{R},\mathbf{t}))^2 + I_{SO(d)}(\mathbf{R}), \qquad (1)$$

where $D_i(\mathbf{R}, \mathbf{t}) = \min_{q \in Q} \|\mathbf{R}_{\mathbf{pi}} + \mathbf{t} - \mathbf{q}\|$ is the distance from the transformed point $\mathbf{R}_{\mathbf{pi}} + \mathbf{t}$ to the target set Q, and $I_{SO(d)}(\cdot)$ is an indicator function for the special orthogonal group SO(d), which requires **R** to be a rotation matrix:

$$I_{SO(d)}(\mathbf{R}) = \begin{cases} 0, & \text{if } \mathbf{R}^T\mathbf{R} = \mathbf{I} \text{ and } \det(\mathbf{R}) = 1, \\ +\infty, & \text{otherwise.} \end{cases} \qquad (2)$$

The ICP algorithm [1] solves this problem using an iterative approach that alternates between the following two steps:

• Correspondence step: find the closest point $\mathbf{q}^{(k)}{}_{\mathbf{i}}$ in Q for each point $\mathbf{p}_\mathbf{i} \in \mathbf{P}$ based on transformation $(\mathbf{R}^{(k)}, \mathbf{t}^{(k)})$

$$\hat{\mathbf{q}}_i^{(k)} = \underset{\mathbf{q}\in Q}{\operatorname{argmin}} \left\| \mathbf{R}^{(k)}\mathbf{p}_i + \mathbf{t}^{(k)} - \mathbf{q} \right\|. \qquad (3)$$

• Alignment step: update the transformation by minimising the l_2 distance between the corresponding points.

$$\begin{aligned}&(\mathbf{R}^{(k+1)}, \mathbf{t}^{(k+1)}) \\ &= \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_{i=1}^{M} \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \hat{\mathbf{q}}_i^{(k)} \right\|^2 + I_{SO(d)}(\mathbf{R}). \qquad (4)\end{aligned}$$

The alignment step can be solved in closed form via SVD [49]. This approach can be considered as a majorization-minimization (MM) algorithm [50] for the problem (1). To minimise a target function f(x), each iteration of the MM algorithm constructs from the current iterate $x^{(k)}$ a surrogate function *g(x | x(k))* that bounds *f(x)* from above, such that:

$$f(x^{(k)}) = g(x^{(k)} \mid x^{(k)}), \text{ and } f(x) \le g(x \mid x^{(k)})\ \forall\, x \ne x^{(k)}. \qquad (5)$$

The surrogate function is minimised to obtain the next iterate.

$$x^{(k+1)} = \underset{x}{\operatorname{argmin}}\, g(x \mid x^{(k)}). \qquad \textbf{(6)}$$

Equations (5) and (6) imply that

$$f(x^{(k+1)}) \leq g(x^{(k+1)} \mid x^{(k)}) \leq g(x^{(k)} \mid x^{(k)}) = f(x^{(k)}).$$

Therefore, the MM algorithm decreases the target function monotonically until it converges to a local minimum. To see that ICP is indeed an MM algorithm, note that the target function for the alignment step is a surrogate function for the target function in problem (1) and satisfies the conditions (5). Specifically, since the closest point $\mathbf{q}^{(k)}_{\mathbf{i}}$ is determined from $\mathbf{R}^{(k)}, \mathbf{t}^{(k)}$, we denote each distance value in (4) as

$$d_i(\mathbf{R}, \mathbf{t} \mid \mathbf{R}^{(k)}, \mathbf{t}^{(k)}) = \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)} \right\|.$$

Then from Eq. (3) and the definition of D_i, we have

$$d_i(\mathbf{R}^{(k)}, \mathbf{t}^{(k)} \mid \mathbf{R}^{(k)}, \mathbf{t}^{(k)}) = D_i(\mathbf{R}^{(k)}, \mathbf{t}^{(k)}).$$

Moreover, from the definition of Di , for any R, t:

$$\begin{aligned} D_i(\mathbf{R}\mathbf{p}_i + \mathbf{t}) &= \min_{\mathbf{q} \in Q} \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}\|^2 \\ &\leq \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \widehat{\mathbf{q}}_i^{(k)} \right\| = d_i(\mathbf{R}, \mathbf{t} \mid \mathbf{R}^{(k)}, \mathbf{t}^{(k)}). \end{aligned}$$

Thus each squared distance term in Eq. (4) is a surrogate function for the corresponding term $(D_i(\mathbf{R}, \mathbf{t}))^2$ in Eq. (1), and the target function in Eq. (4) is a surrogate function for the overall target function in Eq. (1) constructed from $\mathbf{R}^{(k)}$ and $\mathbf{t}^{(k)}$. Therefore, ICP is an MM algorithm that decreases the target function of (1) monotonically until convergence.

APPENDIX II
POSE GRAPH OPTIMISATION

The registration of 3D point sets obtained by LiDARs, RGB-D and stereo cameras is one of the core problems in Robotics and Computer Vision, with applications ranging from dense scene reconstruction to localization. For two point sets, Iterative Closest Point (ICP) [3, 20] is a common procedure to solve the registration problem. This class of algorithms requires an initialization and iterates between estimating point correspondences and computing the optimal transformation between them.

Consider now that we have a set of point clouds corresponding to different views of the same scene. The different point sets may be obtained via an array of 3D scanners. Alternatively, they may represent the visual data acquired by a single observer as it moves through space. Both situations have seen a rise in popularity recently, with applications such as smart-cities and autonomous transportation systems, e.g. self-driving cars and drones, where the registered 3D data may be used to perform object detection, tracking and mapping. For known point correspondences, the rigid transformations that allow for an optimal registration of the point sets can be solved in closed-form via Generalised Procrustes Analysis. For unknown correspondences, optimization strategies analogous to ICP have been proposed. However, such methods are computationally inefficient even for moderately sized registration tasks. To circumvent this issue, Pose Graph Optimization (PGO) is the method of choice in Simultaneous Localization and Mapping (SLAM) and 3D reconstruction.

PGO consists of estimating a set of rigid transformations, or poses, given a subset of pairwise measurements of their ratios. The latter can be computed e.g., by ICP algorithms initialised with 2D image matches. By associating each rigid transformation measurement $\widetilde{\mathbf{M}}_{ij}$, from point cloud i to point cloud j, to an edge (i, j) ∈ E, we obtain a simple graph G = (V, E) which, if connected, can be used to derive a cost function that is minimised by the global transformations Mi , for i ∈ V , that best fit the measured data.

An example of a possible PGO statement is

$$\underset{\{\mathbf{M}_1,\ldots,\mathbf{M}_n\}\in \mathrm{SE}(3)^n}{\arg\min} \sum_{(i,j)\in E} ||\widetilde{\mathbf{M}}_{ij} - \mathbf{M}_i\mathbf{M}_j^{-1}||_F^2$$

In real world applications, guaranteeing globally optimal solutions to PGO is paramount. However, this optimization task is a high-dimensional and non-convex problem. Second-order methods bootstrapped with robust initializations can converge to the sought-after optimum. Nevertheless, they do not scale well. On the other hand, even if certain relaxations of the original problem allow for more efficient implementations, their solutions may be far from those of the original problem

REFERENCES

- Ellon Mendes, Pierrick Koch, Simon Lacroix. ICP-based pose-graph SLAM. International Symposium on Safety, Security and Rescue Robotics (SSRR), Oct 2016, Lausanne, Switzerland. pp.195 - 200, ff10.1109/SSRR.2016.7784298ff. Ffhal-01522248
- F. Dellaert, “Factor Graphs and GTSAM: A Hands-on Introduction,” GT RIM, Tech. Rep. GT-RIM-CP&R-2012-002, Sept 2012
- H. Yang, J. Shi, and L. Carlone, “TEASER: Fast and certifiable point cloud registration,” IEEE Transactions on Robotics, 2020.
- P. J. Besl and N. D. McKay, “A method for registration of 3-d shapes,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 14, no. 2, pp. 239–256, 1992.
- S. Rusinkiewicz and M. Levoy, “Efficient variants of the ICP algorithm,” in 3rd International Conference on 3D Digital Imaging and Modeling (3DIM 2001), 2001, pp. 145–152.