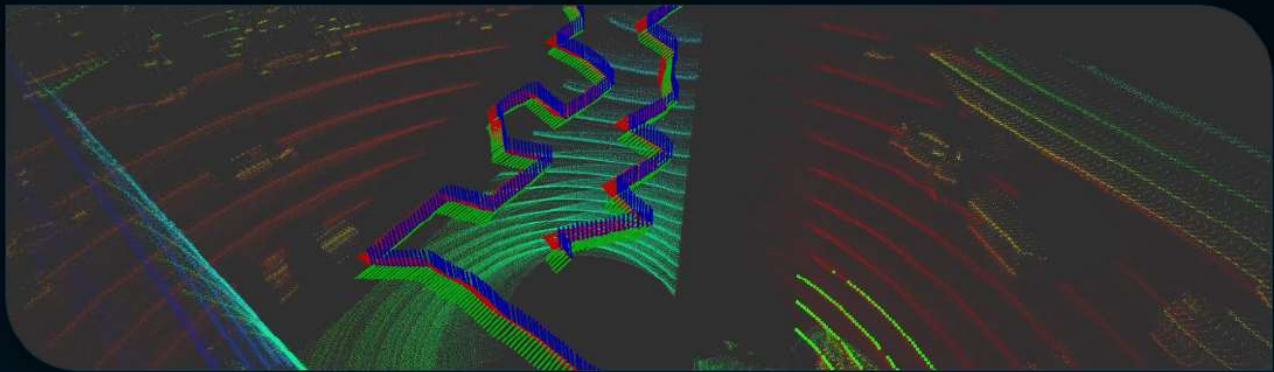


Lidar Odometry: Automated Drones in GPS-denied environments

Blog

2022-05-24



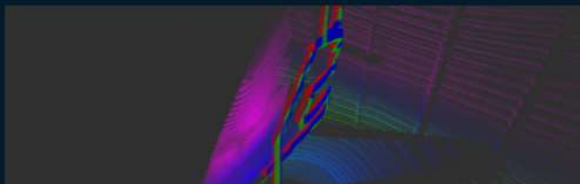
Robots that operate in particularly challenging environments such as aircraft hangars, require fast and accurate state estimation. With poor lighting and no GPS that can be a challenge. That is why the Mainblades development teams are proud to release a new, in-house developed lidar odometry algorithm that has been in daily operation in the maintenance facilities of KLM Engineering & Maintenance for over two months. In this article, [Thomas Horstink](#), Robotics Software Engineer at Mainblades, explains a few technical details and considerations we made to get here.

From Kalman- to Information-filter based Odometry

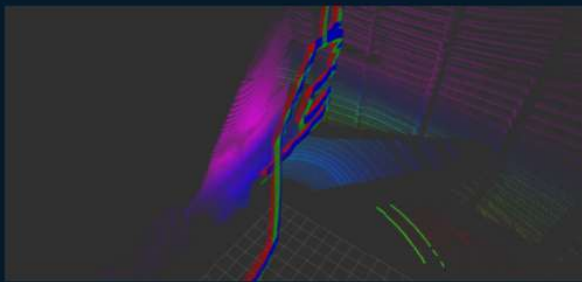
Recently we started using a new internally developed Inertial LiDAR odometry stack for indoor and GPS-denied flights. Before, the drone used a [Kalman filter](#) that combined raw-telemetry with DJI's vision-based system. This setup worked well for a while, but in corner cases, which pop up more often than you expect, DJI's Blackbox odometry algorithm prevented us from flying in certain situations. Hence, we decided it was time to replace the odometry algorithm so that we can guarantee reliable operation in all our use-cases.

The odometry signal is a locally consistent estimate of the drone's whereabouts in the world. It is used as feedback for the controller that flies the drone. It is also used for accumulating observations over time into snapshots of its environment. That functionality allows us to create accurate [3D models of the airplanes](#) we inspect. In the bigger picture, it's an important part that enable a drone to fly automatically.

Crucial decisions for lidar odometry



Crucial decisions for lidar odometry



In the beginning there were two big decisions to be made regarding a new odometry system. Are we going to build an odometry stack from scratch or are we going to build on top of open-sourced implementation? The second choice was about what perceptive sensor to use: lidar, camera or both?

To answer the first question, we experimented with open-source implementations. We set up ROVIO and LIO-SAM. The former uses camera-vision and the latter lidar. The results of both algorithms were impressive and usable with our own data. However, when we judge from a reliability and product perspective, it was more complicated. It is a hard sell to include a few thousand lines of code, sometimes even untested, without thoroughly reviewing and understanding it. It is also naive to assume these implementations meet the requirements we have. When we looked at computational resources, time stamping and signal latency we noticed these implementations were not simple drop-ins. Hence, we decided to take the route of building an odometry stack from scratch.

The second question was easier to answer. During indoor flights we encountered situations in which the lighting was low, and it became troubling for the visual system to keep track. Our main objects of inspection, airplanes, often have large smooth shiny surfaces which can be a challenge to track using a camera. Lidar appeared to have less complications. Which makes sense, and luckily there are few pitch-black aircraft, because that *would* challenge the lidar. But above all, the drone is equipped with a shiny Ouster lidar for model construction and localization purposes. That made our choice for lidar straightforward. For the reason of keeping it simple, we opted to not investigate the combined lidar and camera option unless it was deemed necessary.

Cherry picking

The nice part of creating from scratch is that there are many reference implementations to gain inspiration from. This way, from scratch does not feel like double work, but more like cherry-picking the parts that are needed for our requirements. It is also about deciding on what level we *do* want to depend on open-source software. We figured that writing our own non-linear solver will not be beneficial, nor does it make sense if we implement a scan-matching library. These are examples of components that are well defined, small, and/or clear enough, where we found acceptable to treat them as black boxes.

But when it comes to the logic, models and assumptions, the fabric that turns a collection of measurements into estimates of interest, it pays to be aware of the what's, how's and whys. . And for many small components, we can rely on battle-tested libraries such as GTSAM, PCL, FAST GICP for state-estimation and point cloud manipulation. Within the odometry application itself we use Moodey Camel's lock free queues for internal communication between the 'slow' and 'fast' parts within the algorithm. For interfacing to the rest of our system we use ROS.

The results of the new lidar odometry

The result is a straightforward odometry algorithm that accurately measures the drones' motion. We fuse pre-integrated IMU (Inertial Measurement Unit) measurements with pose-pair-wise scan matches. On top of that we have a tracker that tracks the floor using lidar and adds a prior to each pose where the floor is identified. Robust loss functions are used to mitigate erroneous measurements, and finally the iSAM2-optimizer with a fixed window size is used to combine all the measurements into an (optimal) estimate. To have a low-latency signal for the control loop we use an IMU-based predictor on top of the (slow) lidar odometry.

black boxes.

But when it comes to the logic, models and assumptions, the fabric that turns a collection of measurements into estimates of interest, it pays to be aware of the what's, how's and whys. . And for many small components, we can rely on battle-tested libraries such as GTSAM, PCL, FAST GICP for state-estimation and point cloud manipulation. Within the odometry application itself we use Moodey Camel's lock free queues for internal communication between the 'slow' and 'fast' parts within the algorithm. For interfacing to the rest of our system we use ROS.

The results of the new lidar odometry

The result is a straightforward odometry algorithm that accurately measures the drones' motion: We fuse pre-integrated IMU (Inertial Measurement Unit) measurements with pose-pair-wise scan matches. On top of that we have a tracker that tracks the floor using lidar and adds a prior to each pose where the floor is identified. Robust loss functions are used to mitigate erroneous measurements, and finally the iSAM2-optimizer with a fixed window size is used to combine all the measurements into an (optimal) estimate. To have a low-latency signal for the control loop we use an IMU-based predictor on top of the (slow) lidar odometry.

A blogpost about odometry is not finished without sharing some visuals. Our new odometry algorithm has been in use daily in the [maintenance hangars of KLM](#) for over 2 months now. In the video you can see what the drone is seeing. The big green moving arc is a subset of points that the algorithm assumes to be the ground. The animation is sped up 10 times to save the viewer some time. Have a nice flight!

This article about lidar odometry was written by Thomas Horstink. If you are curious and/or have any questions, feel free to contact him any time via thomas@mainblades.com.

learn more about implementing inspection automation

contact us

main menu

[home](#)

[solutions](#)

[approach](#)

[applications](#)

[about us](#)

[contact](#)

resources

[careers](#)

[news blog](#)

[case studies](#)

[FAQ](#)



Spaarneplein 2
2515 VK, The Hague
The Netherlands



YouTube