University of Glasgow | School of Computing Science

Level 3 Project Case Study Dissertation

CS01 Team Project

Krasimir Ivanov
Mohammad Atbaan Akhtar
Shahzeb Zafar
Shayam Nitin Bhudia
Vrinda Sharma
Krishang Patney

date 2020

Abstract

The abstract goes here

1 Introduction

This paper presents a case study of the development of a sensor mapping software. The application takes data from a LIDAR sensor system like the SOPHIE LITE Electrical External ICD device. The application takes in the scan data and maps it onto a 3D or a 2D map to visualize the scanned data. Furthermore, the mapping function can simulate the sensor system.

The paper begins with some background information about our customer and their proposal (section 2 & 3). It then delves into details about our main application followed by a thorough recount of the software development process (sections 3 & 4).

This is followed by our reflection about the project and key insights gained throughout the six to seven months of development (section 5). We finally conclude it with a brief look at the various software development practices we adopted and how closely our project's progress was tied to the Professional Software Development course (section 6).

2 Our Customer

Thales Group is a French multinational company, founded in December 2000, that designs and builds electrical systems and provides services for the aerospace, defense, transportation, and security markets. The proposal that was given to our team, CS01, by the company's UK branch. Thales UK approached the University of Glasgow Computer Science department to task a student development team to build a proof of concept of sensor data visualization and simulation application.

Our customers consisted of Ian James, Iain Carrie, Matt Tucknott, and Christopher Dickson. Ian acted as our main point of contact and sponsor on behalf of Thales. Iain is the Project Technical Lead for Urban Canyon, one of Thales' many ongoing projects, and was the main beneficiary of our application. Matt is a Thales Technical Expert on GIS (Geographic Information Systems) and has had over 20 years of experience in the field as well as 29 years of Java and C++. Christopher is a Thales Algorithms Engineer with over 19 years of software experience (Python, C++) and has 10 years of experience with image processing (OpenCV, Matlab).

The way our customer's team was structured was that Ian would be our main liaison. Most of our conversations were with him and he was the first to be informed of any issues or scheduling details for meetings from our end. Iain was the one who approved of or suggested many of our application's features earlier on during development while Christopher and Matt provided development support. Matt in particular was integral for fixing one of the major roadblocks we faced over halfway through development (more on this later).

3 The Project Proposal

Thales UK was looking for software to support a LIDAR sensor system with mapping information in 2D and 3D with potential real-time information update capability. Their long-term aim was to bring autonomous and manned platforms closer together with the integrated sensor and mapping software they proposed. The proposed project specification first aimed to develop a simple overlay of a target locator sensor on a map and then aimed to extend the integration to be 3D. The client stated that they were not constraining the development team on what technologies to use, however they would prefer that the mapping tool and the sensor suite be integrated to better aid their aim for the software to cope with a wide variety of situations.

The produced software solution is an open-source MIT licensed prototype, therefore a patent will not be sought. The main purpose and use of the software product are to provide proof of concept. The product will not be deployed into production, only used for demonstration purposes. The goal of the project proposal was to explore the technologies needed to develop such an application, what resources it needs, and to implement a prototype solution.

They stated that their goal for the first couple of iterations was to research available technologies for the framework of the application and what modules to use for the map and the mapping of the sensor information. This was aided by their recommendations on technologies as well as the availability of their software professionals, mapping, and sensing experts. After careful consideration and ranking of frameworks, the team initially settled on using the framework ElectronJS with Cesium, a 3D geospatial modeling tool, to start with.

4 The Application

Before we can discuss the application, we need to know the bare minimum functionality our customers expected of us. At the start of the first sprint, after our first meeting with the customer, we were given three major tasks, with each of them also getting split into a couple of sub-tasks. They were:

Task 1 - Establish a relationship between video and 2d (ground plane) map:

- Task 1a: Populate a 2d radar plot (azimuth vs range) with detections obtained from a video. Assume 2d bounding box in the image, known calibration/pose of the camera, and known distance to object.
- Task 1b: Annotate a 2d ground plane map with detections obtained from a video. As above, but plot on an open-source map based on known vehicle position/orientation.

Task 2 - Establish a relationship between video and 3d map:

- Task 2a: As Task 1b but remove the assumption of known distance to target and plot result on 3d map (or 2d ground projection of 3d map). Distance to target should be automatically determined by intersecting bearing from known bounding box with a ground plane as defined by the 3d map. Report the range to target.
- Task 2b: Project all visible rays from a camera onto 3d map and display total visibility from the current position. Assume knowledge of camera calibration, camera field of view, and camera/vehicle pose.

Task 3 - Plot features from 3d map onto a video feed:

- Task 3a: For an object positioned on the 3d map, determine and display its location in the video feed. Assume known calibration/pose of camera + known vehicle position.
- Task 3b: Combining Tasks 2b and 3a determine whether the object identified in 3d map is visible to the camera and if not highlight that it is a hidden target.

Along with this, it was also requested to make the application open-source. To fulfill our tasks and carry out the request, we were recommended several APIs and software by our customers. Initially, we settled on CesiumJS and ElectronJS but over halfway through the project's duration, we switched to ArcGIS + JavaFX due to the limitations of our earlier approach.

4.1 The CesiumJS + ElectronJS Application

CesiumJS is an open-source JavaScript library for creating static or dynamic 2D and 3D maps. It supports multiple platforms and was perfect for our intended use. We used ElectronJS, an open-source software framework based on Chromium, to create our application and ran the two in tandem on NodeJS which is an open-source JavaScript runtime environment.

This application was extremely versatile. We had a wide range of 2D and 3D map data, provided natively by Cesium, at our disposal and there was extensive documentation available to help us understand and use Cesium's many features. A python script was first used to simulate sensor data and plot points on the map, but this was later scrapped in favour of an Android application we developed which could send actual geographical data gathered from a phone's GPS to the application by utilising WebSockets.

Alas, we soon ran into a few issues which made further development considerably harder, if not impossible. First off, Cesium was very slow. It was not at all noticeable while working with 2D maps but as soon as we made the jump to 3D, we saw considerable slowdowns with the application sometimes not loading the map at all. On top of that, we were unable to project visible rays from the camera for task 2b and the only solution we found involved buying a license for one of Cesium's SDKs which would've meant our application was no longer open-source.

These issues, combined with the application not deploying on anything but Windows and not being able to have unit tests, meant that we had reached the end of the road with Cesium and had to find a new approach to our project.

4.2 The ArcGIS + JavaFX Application

ArcGIS was our answer to Cesium's shortcomings. It is a geographic information system maintained by the Environmental Systems Research Institute (Esri) which can do almost everything Cesium can alongside some of its exclusive capabilities. The application was created on the JavaFX software platform which is cross-platform and supports a wide variety of devices.

The new application was everything we wanted out of our old one. Visible rays from a camera could be projected fairly easily with no additional cost. There is a proper field of view that can be moved around the map, like how a sensor might move, and the performance was better than what we had with the Cesium application. Like before, a python script was used to plot the points onto the map. Sadly the Android application fell out of use due to time constraints as we had to change it to support the new application.

Instead, we were able to implement a few extra features. There's a detailed log of all the points on the map. The points themselves can be saved to a JSON file for backup and loaded again when needed. A script was embedded which simulated a potential path the sensor could take – moving the FOV cone and plotting and logging points as it went.

Unit tests were also added and the entire application itself was added to a CI pipeline successfully. We were not able to do task 3b since we ran out of time but our customers from Thales UK were happy with our progress and took the application from us under the MIT license.

5 Software Development Process

6 Reflection and Insights

7 Conclusions