Multi Sensor Stockpile Measurement – A system for large heap shaped object reconstruction and volume measurement.

BACHELORARBEIT
(Projektpraktikum)

zur Erlangung des akademischen Grades

Bakkalaureus der technischen Wissenschaften

im Bachelorstudium

INFORMATIK

Eingereicht von:
Peter Halbmayer, 9825637

Angefertigt am:
Institut für Pervasive Computing

Betreuung:
Univ.-Prof. Mag. Dr. Alois Ferscha

Mitbetreuung:
Dipl.-Ing. Dr. Simon Vogl

Linz, September 2009
Abstract

Observation and measurement of large scale heap objects, like consumed raw material in iron plants, for deriving volume data from real scale 3D models for automated data processing is done under very special conditions by now.

A complete system is presented, starting at the sufficient acquisition of raw data, over the description of a computational 3D model up to delivering evaluation results that will eventually support business decisions related on more exact base parameters calculated by our system.

The approach presented in this work allows an accurate and automatic extraction of volume measurements for large scale stockpile heap structures by analyzing and evaluating the data of multiple sensor devices.
Contents

1 Introduction 3
   1.1 Physical System Environment .................. 4
   1.2 Software System Overview .................... 5

2 Related Work 8
   2.1 Sensor Data Acquisition ....................... 9
   2.2 3D Surface Reconstruction and Analysis ....... 10

3 System Architecture 12
   3.1 Control Protocol and Data Exchange ............ 14
   3.2 Plugin Architecture .......................... 16
   3.3 State Machine ................................. 17
   3.4 System Configuration .......................... 19
   3.5 Sensor Data Organization ...................... 21

4 Sensor Module 23
   4.1 Sensor Engine .................................. 24
   4.2 Profile measurement ............................ 26
   4.3 Distance measurement ............................ 27

5 Model Module 31
   5.1 Model Properties ............................... 32
   5.2 Model Data Structures ........................... 34
   5.3 Model Module Operation .......................... 35
   5.4 Input Handler ................................. 37
       5.4.1 Sensor Surface Plugin ..................... 37
       5.4.2 Base Surface Plugin ....................... 38
   5.5 Transformation Handler ......................... 38
Chapter 1

Introduction

The measurement of objects in our environment is interesting in many situations. In this work a system is presented that provides three dimensional object reconstruction and volume extraction for large scale stockpile heap structures.

The motivation for this project originates from a practical need of a major Austrian iron and steel producing company to extract the volume of the coal and iron heaps that are on stock to retrieve an overview of the amount of material that was consumed and available on a daily base.

Therefore it is necessary to observe the physical structure of interest with suitable apparatus by which a digital real valued model representation can be generated that gives the possibility to determine structure properties like volume or position of different objects.

The usual way up until now to determine the amount of material that is on stock is by taking inventory twice a year by the application of a 3D lasercanner, with similar technologies applied as mentioned in [34, 18]. Due to the large physical dimensions of the area, it needs to be observed from several positions to retrieve a dense point cloud for the whole area at every position. After data acquisition, resulting point clouds are merged manually to a complete model from which desired properties are extracted. The 3D lasercanner is a rather bulky and heavy device and after data retrieval was ended a separate process chain consisting of model building and manual data extraction needs to be performed. The main purpose of this approach was to have an exact stock enlisting at least yearly to adjust bookkeeping. The whole process of inventory taking needs around one week and is therefore
rather costly and done seldomly. During the rest of the year, dispatcher employees estimate the amount of material on stock. This is very error prone as different employees estimate quite different values. Therefore a way was sought to do an exact structure and volume analysis more regularly to obtain more actual and reliable results.

1.1 Physical System Environment

The application situation is as following. The stockpile area contains a number of heaps where each of them can be of a different material that need to be measured and registered. A crane vehicle, also known as gantry crane, is mounted above the stockpile that is used to unload incoming material trains, restack different materials from one place to another and charge various provision for the production of steel and iron. What is needed by the dispatching department are reports with a detailed enlisting of the amount of material on stock to keep stock records more up to date with more reliable results on a daily base.

To automate data acquisition different sensor devices are applied that can observe the heap structure, similar as done with the 3D laserscanner, but without manual handling of the devices. Therefore 2D laserscanners are mounted on the crane vehicle in a way to record two dimensional cross section profiles of the heaps underneath. By the movement of the crane along the stockpile area all heaps can be recorded. To relate single 2D profiles to each other the position of the crane along the stockpile area is needed which makes the application of a positioning sensor inevitable.

To get a visual clue of the physical area this system is targeting, the image in figure 1.1 shows a photograph of one of the test areas the software was initially implemented for.

The areas that need to be surveyed reach from 400 to 1500 meters in length, about 30 to 50 meters in width and the stockpiles can be an average height of 10 to 20 meters.

Figure 1.1 provides descriptions of interesting spots drawn onto a real world image of the gantry crane. A part of the underlying area with material heap objects can be seen too.
For this purpose an industrial grade computing device (industrial personal computer - IPC) was installed in the cabin of the crane operator from where data acquisition is controlled and volume evaluation can be performed. The laserscanners are mounted in a way to have a clear sight to the surface underneath. They are connected to the IPC by passing through the crane main electrical supply as well as the electrical supply of the crane cabin and eventually reach the computing device in the operator cabin. The GPS receiver necessary for position measurement is installed near the IPC and the required antenna is mounted above the electrical supply container of the operator cabin. More specific details regarding sensor devices are mentioned in the corresponding chapters.

1.2 Software System Overview

Within the typical workflow of the system, it is the responsibility of the crane operator to acquire raw data in a fashion that allows the automated calculation of desired properties. Data is recorded in a session-like style where a single heap, a bunch of heaps or the whole stockpile area at once are observed. The one time acquisition of data is called a scan instance in this context and contains all the sets of raw data from the sensors and related meta data like identification items or sensor configuration for later model processing. Another requirement for the crane operator is to distinguish different heaps with arbitrary material tags so the dispatcher employee can later on relate a heap with the corresponding material the heap is consisting
After a single scan instance is recorded the combination and modelling of the raw data from different data sources has to be performed. The results delivered by the model stage are called *model instance* which contains different 3D models and derived volume data and it can be produced an arbitrary number of times for the same scan instance. The modelling stage has to be designed in a generic way so that different modelling approaches can be applied according to differing requirements of the end users.

An overview volume report is presented to the crane operator so plausibility of results can directly be controlled and the crane operator is enabled in the first course to decide if a scan instance may need to be redone. The dispatcher employee on the other hand is the person who is in response for bookkeeping and ordering materials on stock and therefore needs a more detailed picture of the situation than the crane operator. Due to several reasons like erroneous system handling of the crane driver or sensor failures it might be possible that materials are set incorrectly or that heap boundaries do not correspond to material boundaries and therefore an editing component is needed by the dispatcher where such properties can be adjusted.

![Figure 1.2: General system overview.](image)

According to the user roles and requirements, a general system architecture, as can be found in figure 1.2, was developed. Besides two different end user interfaces two core components were identified, one for data acquisition and a second one to describe an evaluable digital 3D model from raw sensor data. They are depicted in figure 1.2, named *sensor module* and *model*
module respectively.

The roles of the users determine the needed features of the corresponding user interface modules. The already mentioned crane UI serves mainly for the correct acquisition of single scan instances. The application of the model module by the crane UI is needed to provide coarse grained, fast computable modelling results to get validated by the crane operator. The dispatching user interface (dispatch UI) on the other side has to be able to provide more accurate results so they can be used for further business relevant actions. The dispatch UI only needs the functions provided by the model module as it has no direct influence to data acquisition where a detailed description of our approach is given in the user interface section.

The different modules and user interfaces can be deployed in a distributed fashion making the system suitable for operation in distributed environments. Although it is possible that data acquisition is controlled remotely the most practical approach is to host the crane UI and the sensor module on the same machine. The model module and the dispatch UI on the other side can be distributed over different machines with the only constraint of a network connection between hosts. Because modelling is done in an offline mode, a scan instance, after it is fully recorded, may be requested at any time from the sensor module to decouple model processing from data acquisition, which can then be done on a more powerful computer for example. This way a flexible layout for deployment can be specified and will vary between different system installations.
Chapter 2

Related Work

A full system description needed to be developed for the practical project work and therefore the underlying software system has a strong focus to corresponding practical issues. Related projects often implement lower layers of the software architecture in a static way which allows a low interoperability of different subsystems within alternative environments. One major challenge of this work was to create a system that can be applied within varying environmental conditions to be easily adapted to other measurement tasks for the purpose of the automation of these. This software architecture centered view is hardly found in related work and provides common procedures when dealing with such systems.

Where the differentiation between sensor and modelling issues serves well for software architecture considerations it is obvious that one part does not make sense without the other. Most related work is based on this fact and either presents some static sensor deployment or assumes preknown sensor raw data. The environment, this work was verified in, is also determined by a static installation setup but the software was developed in a way to replace different subsystems easily to support alternative application scenarios.

In related projects, sensor systems are laid out to fit the 3D modelling process whereas this work tries to give an entire picture of all the processes involved for handling large scale volume measurement tasks automatically. A recent and detailed discussion of all the processes involved in acquiring 3D laserscanner data, transforming it to a computational representation up to visualizing and evaluating resulting 3D models is given in [32].

Actually laserscanners are the most widely used devices to accomplish 3D
reconstruction of digital representations of the physical objects like the modelling of statues or archeological relics (cf. [22, 23]), registration of indoor environments (cf. [29]) or terrain and building modelling (cf. [3]). Alternative applications of laser sensors were found in the automotive industries (cf. [19, 16]) or in robotics (cf. [6, 2]).

2.1 Sensor Data Acquisition

The provision of sensor technology for object reconstruction is a research area of its own and not further treated in this work as it is more a matter of physics or optics. Industrial companies already provide a wide variety of sensor devices as well as accompanying software systems tailored to specific application needs (cf. [14, 1]).

For 3D object reconstruction laser scanning systems are applied where two different device constellations are common.

For industrial applications, devices are used (as in this project) that provide a laser beam, that is invisible to the human eye (infrared), which is targetted onto a point of a surface and the time of flight is measured until the reflection of the beam is received. The beamer and receiver are mounted within the same line of sight over a sophisticated installation of mirrors so they are contained within a single physical device. From the time of flight the distance of the device to the first point of a surface is calculated. If no reflection is received an invalid value will get output. By putting the laser projector onto a device rotating in one or two axes a 2D or 3D lasercaner is obtained. They produce 2D and 3D profile data given as semi structured point clouds. Applications of such devices can be found in [2]. A 2D lasercaner can be turned to 3D scanner by keeping it moving along the static axis as is done for example in [34].

The data from a lasercaner is received in a polar coordinate format where the center of the device represents the origin of the coordinate system. To utilize the lasercaner data it has to be transformed to an euclidean coordinate space in which usual 3D processing algorithms are commonly available.

In laboratory environments often an alternative approach is followed because the above mentioned lasercaner devices are often bulky and designed for outdoor use. For indoor scenarios two different devices are used to achieve
similar results as can be found in [8] or [4]. A laser beamer usually projects a whole visible line onto the surface of the physical object to be measured while a CCD camera records the skimming of the laser beam over the object. To retrieve depth data from continuous 2D images the geometric relation between the laser beamer, the camera and the observed object must be given which is often called triangulation in this context because of the three physical objects involved that describe the corner points of a triangle. This can often be a source of errors that are reduced by applying different estimations in the modelling stage.

An interesting fact about both types of measurement setups is that their results can be represented as semi structured collections of geometric points, given within some reference system, that is provided by both approaches. Therefore the methods and algorithms used by the two approaches can be applied interchangelably.

An emerging trend is the combination of different sensor devices which is one of the core questions of this work. By the time of this writing, 3D laserscanner products are existing that have an integrated GPS receiver to determine the global position of the scanner in a world coordinate system as is applied in [12]. Because the devices usually are handled from a fixed location within a period of time, the position of the device does not change within a scanning session (besides some noise related to the technical characteristics of GPS) and therefore all acquired data needs to be related to this certain point. In our approach the application of 2D laserscanners is sufficient because the application of an independent, mobile GPS receiver unit allows for determining the exact position of every single 2D measurement at every instance of time.

In this work a set of sensor devices is installed at locations from where informal preliminary considerations indicated that this combination of devices will provide the most promising results for acquisition of raw data to produce appropriate 3D models.

### 2.2 3D Surface Reconstruction and Analysis

The reconstruction of three dimensional objects whose raw data is available as a point cloud of 3D depth information is mainly based on the definition of a model that transforms the acquired raw data into a scalable digital object that
can be analysed under different conditions and requirements. The central data object to the modelling stage is the 3D surface or surface mesh that is a collection of points in a common coordinate system or that can be transformed from one coordinate system representation to another. Surface meshes can be processed with a large amount of available algorithms to extract different results like visualizations or characteristic values of 3D objects. Therefore methods like edge extraction (cf. [20]) or geometrical operations (cf. [25, 24]) are applied as well as boolean operators (cf. [27]) or feature extraction (cf. [9, 10]).

To obtain a closed surface from a point cloud the single 3D vertices have to be connected in one way or another and results in computer graphics have shown that a representation as a set of connected triangles is very convenient for such purposes. One of the most prominent methods is Delaunay triangulation\(^1\) (cf. [28]) that produces two dimensional meshes that is minimal in the size of the produced triangles and allows geometrical analysis of the considered surface (cf. [15]). In this work an adaption of the modelling approach applied in [14] is used as well as Delaunay triangulation. Alternative methods for surface reconstruction are discussed in [32].

Arbitrary filters can be applied to raw data as well as during different model processing stages to reduce the error rate given by outliers or general measurement failures caused by physical hardware restrictions (cf. [26, 31]). Hardware device specific filters are treated within the sensor module and on a 3D object level special functionality is implemented regarding filtering.

More general work that addresses problems of 3D reconstruction and modelling of large point clouds can be found in [5] and [33]. The analysis and visualization of stockpile areas is also treated in [21], where approaches of the classical terrain measurement are applied.

\(^1\)Here a point set triangulation is meant whereas the triangulation above has a different meaning. The mathematical principles are applied equally.
Chapter 3

System Architecture

A general overview of the system architecture has already been given in the introduction section whereas this section presents a more thorough picture discussing core system components in more detail. The different subsystems were identified by analyzing the flow of data and the tasks that have to be accomplished to retrieve the real valued volume of a physical object and the belonging 3D model from real time constrained point clouds given in polar coordinates and a set of position values from a corresponding position measurement source.

Figure 3.1: Multi sensor stockpile measurement system architecture.
In figure 3.1 the same schematic is found as in figure 1.2 this time with more detail revealed. The sensor and model modules form the core of the system and are developed in a similar fashion to unify access to components and keep system design simple. The blue lines with a forward and backward arc in figure 3.1 show the direction of control (denoting requests and corresponding responses) whereas the red lines with a single arc in one direction denote the flow of data.

The server components that can be found with every module are the main entry points of the modules for the outside world and support a set of commands to retrieve information from single subcomponents and the modules as a whole. A simple protocol was implemented that is capable of transmitting and interpreting command requests as well as file data by applying the HTTP protocol. Commands from the clients are always forwarded to the corresponding engine objects that interpret the command and trigger according action. The engine components define interfaces that are used by the server entities to control the engines as a whole or to address single subcomponents of every engine.

Requests to the model server can come from two different user interfaces and therefore different results are needed according to the requirements of every interface. The crane UI expects only a coarse summary of the volume measurement of observed objects whereas the dispatch UI needs a more detailed account of every model instance.

Because of this webservice approach to connect to the system modules, it is possible to specify user interfaces that are simply accessible over a web browser. To support special features like online scan views, additional access channels are defined that are not depicted in the diagram as the main system operation does not make use of these features.

The crane UI is the component where data acquisition accomplished by the sensor module gets triggered. It controls starting and stopping scan instances and displays realtime data from the different sensor devices as an online feedback to the user. Each scan instance is registered with a unique identifier by which it can be located within the system by depending components. When a scan instance is recorded, the configured set of sensors measure the heap objects under consideration while the crane operator moves the crane vehicle along the interesting part of the stockpile area. The continuous streams of observation data from the sensor devices are recorded and saved
in the sensor data store and can later be accessed by directing corresponding requests to the sensor module.

After a scan instance has been recorded, it can be accessed by the model module for 3D structure reconstruction and derivation of volume measurements. The ID by which a scan instance was registered is passed to the model module that is needed to request the corresponding data from the sensor module. After the configured modelling tasks have been performed a model instance, containing resulting data of this stage, is saved to the model data store the same way as is done by the sensor module for scan instances. This way, data that is output by the model module can be accessed in the same style over the model server.

3.1 Control Protocol and Data Exchange

To exchange commands and data between components, a simple protocol was developed that is an adaption of the REST protocol that is described in [13]. A description of the different messages applied between the modules is given by a sequence diagram that shows the message exchange until the first model evaluation results are available for the common system workflow. The diagram can be found in figure 3.2.

It can be seen that the processes for data acquisition and modelling can be clearly separated where only one of the two modules is concerned in the corresponding stage. This makes it possible to decouple system components and apply them for completely different requirements.

For a typical program run, the crane operator starts a measurement session by sending a start command to the sensor module over the crane UI. As an identification for the scan instance that will get created, the timestamp when the start command is issued, is used. The sensor module allocates resources needed for this scan instance, starts all configured sensors and appends newly acquired data to the scan instance. During this part the crane operator steers the gantry crane along the measured object in an appropriate fashion to obtain data useful for modelling. Additionally, a heap object demarcator needs to be set whenever different material appears below the laserscanners. While the sensor module is recording the actual scan instance, the crane UI requests the status of the module to check that configured sensors and higher level components are operating as expected.
When the actually interesting area has been scanned, the crane operator commands the sensor module to stop recording of the scan instance and implicitly initiates operation of the model module by issuing a start request with the identification of the scan instance. The model module requests the data of the scan instance from the sensor module and starts processing of the data after it is received according to the configuration of the module. Meanwhile, the crane UI polls the model module to get informed about the progress of 3D model processing. This is done until modelling has finished and an idle state is reported by the module containing results about the last evaluated scan instance.

In table 3.1 example request URLs are depicted that control the most important functionality of the modules. After the host and port part of the request a path like expression points to the component that needs to be controlled. This can be a single subcomponent of a module or the module as a whole. An example can be found with the status command of the sensor module where in the first case the status of the whole module is requested whereas in the second case only the state of a single sensor is queried. The responses returned to different requests are not shown in the table as only
standard HTTP responses are used, containing the requested status or if the command was executed successfully or not in their payload.

The webservers of the modules run on different ports therefore a single PC can host both components simultaneously and the command format is almost equal for the two modules. After the command target a list of parameters is appended that contain different properties like the specific command or a date field that serves as an identification for scan instances. The format property is mainly used for testing purposes as the servers are capable to service a simple web interface as well as the specialized interfaces used in our approach.

<table>
<thead>
<tr>
<th>Module</th>
<th>Command</th>
<th>Request</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>stop</td>
<td><a href="http://localhost:5000/system?cmd=stop&amp;format=text">http://localhost:5000/system?cmd=stop&amp;format=text</a></td>
</tr>
<tr>
<td></td>
<td>status</td>
<td><a href="http://localhost:5000/system?cmd=status&amp;format=text">http://localhost:5000/system?cmd=status&amp;format=text</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://localhost:5000/sensor/s1?cmd=status&amp;format=text">http://localhost:5000/sensor/s1?cmd=status&amp;format=text</a></td>
</tr>
<tr>
<td></td>
<td>stop</td>
<td><a href="http://localhost:5001/system?cmd=stop&amp;format=text">http://localhost:5001/system?cmd=stop&amp;format=text</a></td>
</tr>
<tr>
<td></td>
<td>status</td>
<td><a href="http://localhost:5001/system?cmd=status">http://localhost:5001/system?cmd=status</a></td>
</tr>
</tbody>
</table>

Table 3.1: Example control commands.

### 3.2 Plugin Architecture

As can be seen from figure 3.1, every module contains an engine component that is in care of managing the core functionality of the module. For this purpose, a plugin architecture has been developed that is applied by the engine instances to define the way in which a module accomplishes its tasks.
In case of the sensor module, the sensor engine directly accesses the sensor devices over plugin wrappers that support a common interface. Essentially a sensor only needs to support commands to start and stop the device, a reference to a place where recorded data has to be stored and a way to request the status of a sensor to control the ability to operate successfully.

The model engine on the other side needs an additional level of abstraction represented by the plugin handlers. For the purpose of this work three different handler objects are needed: The \textit{input handler} cares for acquisition of input data from various sources, the \textit{transformation handler} converts the input data to the internal format of the module and the \textit{model handler} eventually achieves the model evaluation and volume extraction calculations. In case of the input handler for instance, incoming data can be of various formats and it is convenient to summarize different input functions under a common handling resource. The actual handling of a specific resource or model interpretation is done by plugins implementing a special format or task. This way the same scan instance can easily be modelled using different reconstruction approaches to achieve comparable 3D models for optimization of the modelling task.

\section*{3.3 State Machine}

Because plugins may apply logic that runs within its own control loop (and certainly does for the sensor devices) decoupled from the main control loop of the module, a way needed to be found to ensure the responsiveness and revise the liveliness of every module and belonging subcomponents. For this purpose a state machine was established to which every module and components applied by the modules adhere, in order to treat runtime tasks and error situations in an equal fashion.

Every part of the system core provides an implementation of the state machine on its own, where the state of a higher level component may be derived from the states of associated subcomponents. This is handled a bit differently by the sensor module and the model module and is mentioned in the corresponding sections. Here a general interpretation of the different states and the transitions between is given and the corresponding diagram is shown in figure 3.3.
When a component is instantiated for the first time, the init transition is executed that cares for initialization of the corresponding part. Resources that need to be available for the whole lifetime of the component are set up like connection to sensors or pointers to toplevel data storage entries.

In the idle state the corresponding module waits for any user requests to start its operation. It occasionally investigates the status of connected subsystems and switches to the error state if a subsystem reports any failure or is unreachable. It is possible that the idle state is reentered from an error state if the failed component is able to recover itself and the system can go back to normal operation.

The switch from the idle to the running state happens explicitly upon user request or implicitly because a higher level component has switched to the running state and is applying the functionality of the subcomponent. The request to stop a runtime task is handled the same way too, at least in the case of parts that run in their own control loop. For sequential logic that is executed only once the transition between idle, running and back is done automatically.

The error state is needed to catch any special conditions that hinder the system from normal operation. Usually, the system is polled to retrieve if it is still functional in the idle state or achieving the task that has been triggered recently. If some error condition arises the component tries to recover itself and switches back to the idle state in case of success. If for example a sensor device is unreachable the wrapping logic switches to the error state and continuously polls the device until it becomes available again. This way the system is still usable although not fully functional. If error recovery of any subsystem is not possible the user needs to get notified and special maintenance action has to be taken to reactivate the component.

Switching back to running from the error state makes no sense as in-
formation vital for the running state may get lost or deleted by the failure occurrence like incorrectly terminated data files or invalid time stamps.

Not every component provides an implementation for every state especially when the component does not make use of an internal thread. In this case only a placeholder for the state is implemented to keep the call interface consistent.

3.4 System Configuration

In the recent implementation of the system, settings of the modules are handled over configuration files that are found in a dedicated filesystem location of the hosting computer. The sensor and model modules can be configured separately to handle the modules in an independent way which is necessary when the modules are deployed on different hosts. Single configurable items are set over key, value properties and only sensor devices and model module plugins are configured by nested properties.

An example for the sensor module configuration is given in listing 3.4. First in the configuration file, the host and port of the web server are noted to determine where the service of the module is provided. Afterwards the settings for one lasercanner and a single GPS receiver are stated. In our installations two lasercanners are deployed and the configuration for additional lasercanners are basically the same as for the first one. They only differ in their IDs and parameter values according to their physical location. The specific instance of the sensor type is determined over the type field within the sensor configuration. Depending on the type of sensor different structures are allocated.

Lasercanners are connected over a static network therefore an IP address and a port number are given, by which the corresponding devices can be reached. The start and stop angles, the scan resolution and the motor speed set the operation parameters of the device. They are implicitly contained in the data output by the scanner. The stream port specifies where online data for the crane UI can be acquired. This field is present for every sensor. The following properties determine the transformation parameters from the local lasercanner coordinate system to a global model coordinate system needed by the model module and they are set within the scan instance meta configuration.
The GPS receiver is the second type of sensor device that needs to be configured for use by the sensor module. Because it uses the gpsd\(^1\) package that connects to the GPS device over a serial link and provides its service over an IP socket to the outer world. The start and end longitude and latitude are needed for relative distance measurements and are described in the corresponding section below. The stream port is the same as in the case of a laser scanner device.

```python
sensor_host = "localhost"
sensor_port = 5000
sensor s1(
    type = "ls"
    ls_host = "192.168.1.10", ls_port = 49152
    sector_start_ang = 142.50, sector_stop_ang = 228.75
    scan_res = 0.25, motor_speed = 10
    stream_port = 5024
    trans_x = 14.87, trans_y = 0.0, trans_z = 31.19
    scale_x = 1.0, scale_y = 1.0, scale_z = 1.0
    rot_x = 0.0, rot_y = 1.0, rot_z = 0.0, rot_ang = -90.0
)

sensor p1(
    type = "gps"
    gpsd_host = "localhost", gpsd_port = "2947"
    start_lon = 14.32972, start_lat = 48.27832
    end_lon = 14.33357, end_lat = 48.27884
    stream_port = 5025
)
```

Figure 3.4: Sensor Module configuration listing.

The model module the configuration is somewhat simpler than in the case of the sensor module, as only a small amount of information must be available prior to the first start of the module and missing information is passed from the sensor module within the exchanged scan instance. It can be found in listing 3.5.

As for the sensor module the host and port where the service of the module is provided, need to be set. Under the plugin handler property several relative locations are stated, where plugins can be found. This is described in more detail in the corresponding section. The connection properties for the sensor host and the base file that follow in the configuration may be set

\(^1\)http://gpsd.berlios.de/
model_host = "localhost"
model_port = 5001
sensor_host = "localhost"
sensor_port = 5000
base_file = "gelaende/k310/urgelaende_k310_final.dxf"
cell_size_x = 1.0
cell_size_y = 1.0
res = 0.1 #10 cm
avg_width = 8.58
cut_val = 5.95
wall_cnt = 30
bunker_volumes = {52.0, 96.0, 138.0, 170.0, 202.0, 224.0, 244.0, 258.0, 270.0, 282.0, 316.0, 350.0}
plugin_handler = ("input", "transformation", "model")

Figure 3.5: Model Module configuration listing.

within the input plugin directly but is stated in the module configuration file to keep settings simple and consistent. Their usage is described when reaching the corresponding plugin.

3.5 Sensor Data Organization

Figure 3.6: Scan Instance Diagram.

Although specific properties of single sensor devices are first discussed in the next chapter an overview of the data they produce is already given in this place, as it strongly relates to the sensor configuration described above.

An example scan instance is shown in figure 3.6 where a configuration of two laserscanners and a GPS position measuring device is applied as was the case in our test environment. A meta configuration file determines the settings for this scan instance and describes the sensor entities that participated in this scan instance. Every sensor holds a pointer to a scan set, that contains the raw data generated by the sensor. Each scan set is of a
specialized subtype according to the type of sensor that created the data set.

In case of the laserscanners every data set consists of a list of scan slices where a single scan slice represents the measurement of a lasercanner for a single measurement. The GPS sensor records a single list of position values relative to an arbitrary origin annotated with a timestamp when the position measurement occurred. The origin is chosen to reflect one of the corner points of the stockpile area and the position values can be adjusted by a correction offset parameter.
Chapter 4

Sensor Module

To extract the volume of three dimensional structures it is first necessary to acquire data in a fashion that is suitable for model creation and evaluation. The Sensor Module connects to different sensor devices that record raw data in a form that allows successful model construction and evaluation, which is described in the following paragraphs. Because heap object structures with a large, real world scale need to be observed and represented, the types of sensors used for data acquisition have to register environments with an accuracy hardly found in consumer grade devices. The different sensor devices applied in this work and their specific properties are described in the following sections.

The sensor server accomplishes the connection to the outside world and provides a protocol that allows working with the module over a web interface. The sensor engine is responsible for managing a set of configured sensor devices to achieve data acquisition in the desired manner. The particular sensor logic provide the direct connection to the single sensor devices and implement the interface by which the sensor can be controlled by the sensor engine. At the moment two different types of sensors are used to record a dense point cloud of the area that needs to be surveyed. They are described in the following paragraphs.

The system is designed in a way so that an arbitrary number of sensors can be deployed. This depends on the requirements of the site to be surveyed and needs to be adjusted for every installation separately. In this work the installation of the first prototype system is used as the base for the description found herein, but the number of sensors would not change the
general work flow of the core sensor system.

Sensor data is saved in a preformatted raw form with basic filters applied that mainly reject values due to physical constraints of the device, for example when the crane vehicle temporarily stops moving while recording a scan instance.

In this paragraph the advantages of using multiple laserscanner devices are described whereas the internal properties of a laserscanner are mentioned in the corresponding section. When a single laserscanner device is mounted near the center of the crane vehicle, situations may arise where the heap objects are not overseable at all places. A schematic of this scenario is depicted in figure 4.1a. Here it can be seen that the scanner delivers a sparse point cloud for areas of certain form and that a part of the heap can not be seen at all. To overcome this issue the approach depicted in figure 4.1b is chosen: By the introduction of more than one laserscanner, areas that are problematic with a single scanner can be observed in the way required. The number of laserscanners deployed or the density of acquired data points is irrelevant to the sensor module and is handled within the model module.

4.1 Sensor Engine

The sensor engine is responsible for the management of connected sensor devices and forwarding commands from the sensor server to the corresponding sensors. Acquisition of scan instances is handled over the sensor engine. When a request to start a new scan instance is received, a scan instance according to the passed timestamp is created in data storage and all configured sensors are started sequentially. A sensor device runs in its own thread of execution and writes data to an exclusive location (unseen by other sensors) within the scan instance. As every sensor is able to run independently of the others the sensor engine must take care for correct termination of a scan instance and that all resources occupied by the sensors are released in an organized way.

The state of the sensor module is determined by the sensor engine which represents the states of single sensors within a single compound state. Normally all sensors are in the idle or running state and if not, the error state is entered. An initialized but nonfunctional sensor puts the whole module
in the error state, too. This way, under normal operation conditions, the sensor module simply switches between the idle and running state.

Before an explanation of the properties of special types of sensors is given the actual connection to the different sensors need to be mentioned. Physical sensor devices are connected to the controlling computer device mainly over serial or networked connections. These connections are implemented as wrappers for the different types of sensors that support a common interface. Therefore different sensors can be addressed the same way hiding implementation details to a single part directly related to the sensor.

Connected sensors are determined over a configuration file that contains all information about a single sensor needed within the overall system.
Filters that may be applied on the data of the sensor are specific to the type of sensor and not to be confused with filters in the model module that operate on other types of datastructures. The application of single sensor filters is treated below.

4.2 Profile measurement

The first kind of sensors applied in our system are 2D laserscanners of the type SICK LD-LRS3100 (cf. [1]). This devices measure continuous two-dimensional profiles applying a time of flight infrared laser technology. Within the device a stepper motor is installed that drives the movement of a laser light beam generator. The distance of every light pulse to the first contact surface is measured by the round trip travel time of the pulse. The center of the device serves as the origin of the coordinate system local to the scanner. As the rotating light beam generator is mounted on this origin an angle for every light beam can be defined. The stepper motor is configured to step through a range of angles and all distances between this range are recorded several times a second. The laserscanners have to be mounted head over and the interpretation of the 2D coordinate system by the scanner device can be seen in figure 4.2. The laser beam can record distances up to 250 meters. The range of angles that are observed by a laser scanner have to be configured for every device separately according to the physical location where the device is mounted on the gantry crane and lies around 90 degrees. The angles of a laser scanner are set up to have a quarter degree resolution to obtain the highest possible resolution the device offers, which results in 360 acquisition points contained within a single scan slice for a 90 degree scan width. With a frequency of 10 Hz, 3600 data points are generated per second.
One measurement from a laserscanner is given as a single pass through the configured range and contains a list of distance values with the corresponding angles and is called a scan slice. The movement of the crane vehicle along the stockpile area produces a continuous stream of scan slices given in the local polar coordinate system of the laserscanner that are independent of each other. Only by the application of another type of sensor, a fully connected 3D representation can be established.

As the acquisition of a single scan slice is done by the laserscanner internally and can not be interfered by user software, a timestamp at the start of the measurement and another one at the end is recorded too. These timestamps are needed afterwards to relate the data from a single laserscanner to data from the other sensors.

Because the laserscanner devices need to operate in an outdoor industrial environment they are prone to get dusty over time limiting their ability to work correctly. The internal filter implemented for the laserscanner now targets this problem as it will calculate a value that indicates the degree of pollution and can be used to inform the user if the device needs to be cleaned manually.

4.3 Distance measurement

Besides successive two dimensional profile data from an observed object a position measurement is necessary to determine the right three dimensional expansion of the object. In related projects, the distance between scan slices
is estimated according to the movement speed of the hosting device where
the scanner is mounted or is given by some measurement device constant.

The third coordinate that is needed to build up a three dimensional
structure representation is given by a relative position of the crane vehicle
along the stockpile area. Initially the Jenoptik JENO LDM301 distance
measurement device was applied (cf. [17]) but was rejected in favor to a more
practical solution over GPS mainly due to construction related reasons.

As the LDM301 provided single successive position measurement values,
annotated with the timestamp when the measurement was accomplished,
a similar output was expected from the GPS sensor so depending system
components need not be modified (this mainly applies to the corresponding
input plugin of the model engine). Moreover does this approach produce less
raw data than storing complete GPS NMEA sentences. A transformation
between GPS NMEA data to single position values needs to be accomplished
to receive the desired format.

The transformation process can be described as depicted in figure 4.3a.
Here a sky view representation of the stockpile is given which can be ap-
proximated as a rectangle with the starting point in the upper left corner
available as a longitude, latitude pair \((lon_{\text{start}}, lat_{\text{start}})\). First a line has to
be defined upon which the position of the crane can be determined and this
is possible by configuring values for \(lon_{\text{end}}\) and \(lat_{\text{end}}\). The start and end
positions (given in GPS coordinates) indicate the track that the gantry crane
is able to observe. Now a measured coordinate pair \((lon_{\text{meas}}, lat_{\text{meas}})\) can be
projected onto this line yielding \(lon'_{\text{meas}}\) and \(lat'_{\text{meas}}\). This way the distance
of the projection to the start point can easily be calculated which is eventu-
ally necessary to serve as third dimension for modelling. The application of
the Haversine formula (cf. [30]) gives even more accurate results as it takes
spherical geometry into account and considering the elliptical form of the
earth increases accuracy even more, but the principle stays the same.
Figure 4.3b shows a satellite view of a stockpile area annotated with the raw GPS coordinates measured during a drive over most of the area (shown by the blue line in the figure). By the thickness of the line it can be seen that the GPS measurements do not describe a straight line but drift around the main movement line. How to overcome this noise recorded in the measurements is discussed below.

In the case of our distance measurement approach it is possible that when the crane vehicle performs a constant movement in one direction that sampling points will indicate a movement in the reverse direction, mainly caused by measurement errors of GPS. This is avoided by the application of a simple median filter where the median of a number of latest measured values is taken as the value of the current measurement. By defining the filter window reasonably large, even bursts of outliers can be eliminated. It is also possible, due to temporal physical occlusion of the antenna, that no sensible values are received which also need to be disregarded.

With the above calculations a one dimensional stream of distance values is received that are annotated with the time of their measurement to be related to corresponding scan slices within the same scan instance. It can be seen that the operation of the position measurement is completely independent of the profile measurement sensors. To obtain the necessary accuracy for the distance measurement the SPS550 Modular GPS Receiver produced by Trimble (cf. [11]) was used. The measurement frequency of the GPS device is also 10 Hz but this time producing 10 single values augmented with their timestamp of occurrence are collected. Therefore the amount of
data gathered by distance measurement is far less than in the case of the laserscanners.
Chapter 5

Model Module

The model module is the part of the system where the assembly of a geometrical 3D model from the raw sensor data and an evaluation and extraction of object volume takes place. Within the typical system workflow the functionality of this module is triggered when the user terminates a particular scanning procedure. It gets passed the identification of the scan instance that needs to be modelled, by which the corresponding data is retrieved from the sensor server. After converting the raw data to an internal format, model specific transformations are applied and a corresponding 3D model is produced from which volume measurements can be derived. Output data resulting from the previous steps is again put in a data storage location and can afterwards be accessed the same way as data from the sensor module.

In the same fashion as the sensor module, the main components of this module consist of the model server and the model engine with connected handler objects and installed plugins. Different methods of modelling can be applied on a single scan instance. Therefore the module is highly configurable to evaluate different modelling approaches against each other where an example configuration was already shown in the system architecture section. This flexibility is reached by the definition of different handlers that organize various plugins needed for modelling of stockpile measurements. Another advantage of this generic approach is that modelling challenges from other domains can be implemented easily.

The measurement results as required by our customer contain a list of material heaps, material types and ranges, and the extracted volume on a per heap base. Accompanying with any graphical output used for visual
evaluation this data describes a model instance that is the final output of the model module which can be used for further decision directions. A model instance contains all information that is needed to extract the amount of material on stock for a certain date in time.

5.1 Model Properties

Before the needed data structures and the different functionality of the model module are discussed some general terms need to be defined that provide the basis for the further discussion.

Initially, the ground surface is unknown to the scanner devices as it cannot be expected that the area will ever be observed in an empty state where no material is on stock for a ground reference measurement. Therefore, a base model is used that represents an idealization of the stockpile area in an empty state which is provided by the civil measurement project partner company and an isometric view is shown in figure 5.1 with the coordinate axes depicted as they are interpreted throughout the modelling process. It represents the base surface that is used in the prototype installation of the software system.

Basically the modelling and volume extraction process achieves a comparison between the surface as recorded by the sensor devices and the reference base surface where the space between the two surfaces determines the amount of volume for the considered heap object. The bounding box of a surface, that describes the minimum and maximum extent of a surface in any direction, is a structure to achieve simple surface operations. It is needed throughout the whole modelling process.

The bottom left corner of the bounding box of the base surface is selected to serve as reference location for the global model. All models given in a local coordinate system have to be transformed to this common origin. By the application of the distance measurement onto the profile measurements the exact position where the heap object is located above the base surface is found.

From figure 5.1 it can be seen that the base surface consists of two structurally different parts. The plane part is the main area of the stockpile where different material heaps are filled up and it is simply called the stockpile surface. The comblike structure at the upper edge of the area is used to bring
different material to the stockpile and every chamber within this area is called a \textit{bunker}. The whole part of the bunkers within the entire model is called the \textit{bunker surface}.

The dimensions of the bounding box of the stockpile surface used within the prototype installation example are 54 meters width, 424 meters length and 12 meters height, where the height is determined by the height of the bunker part of the surface. Without bunkers the height reaches around 2 meters. The height of heap objects can reach 25 meters which is the maximum distance between the base stockpile surface and the lower edge of the shovel of the gantry crane that is used to fill up the heap objects.

To support the understanding of the role of the bunker structures their usage is explained in the following. They provide the mechanical feature to deliver material charges to the stockpile. Material trains (that are not depicted in the figure) move along the top of the bunker area to empty their contents into the bunkers. From there the material can be reached by the shovel of the gantry crane by which the stockpiles are heaped up. It is possible that material heaps become that large to fill part or all of a bunker and therefore it was an explicit requirement to evaluate the volume contained within the bunkers.

It has shown practical to define several terms dependent on the perspective from where the stockpile area is viewed. The \textit{top view} is the view that the software uses throughout the 3D modelling process where the stockpile area seen from an above perspective, with the width of the surface interpreted as $x$ coordinate, the length as $y$ coordinate and the height as $z$ coordinate.
and the looking direction is alongside the negative $z$ axis.

The front view is the view looking alongside the $y$ axis of the stockpile area and by the physical orientation of the laserscanner sensors front view profiles, given in polar coordinates local to each sensor, are generated. A sequence of front views along with the distance measurement eventually determine the 3D model that is used for volume extraction.

Finally, the side view is needed when presenting a summary report of the model evaluation, where heap structures are seen from a side perspective looking over the stockpile area into the direction of the bunkers along the $x$ axis. This view is convenient to present an overview of the of the stockpile where important values can be included in an easy recognizable way. Closely related to the side view is the meaning of material ranges that are used to delimit heap objects consisting of different materials. A material range defines from which starting point to which end point a heap of a certain material reaches, the specific material it comprises of and the volume of the heap object. Because a scan instance can include more than a single observed heap object a list of material ranges must be maintained and an example is shown when the user interfaces are discussed, as the material range list is one of the main results that needs to be delivered to the user. A single material range reaches from the first scan slice of a heap structure until the last and it is set by the crane operator. If later it is recognized, that the material ranges are set incorrectly they can be adjusted as needed by the dispatch user.

5.2 Model Data Structures

As internal format for modelling data structures from the GNU Triangulation surface (GTS) library are reused \(^1\) that are extended with self defined fields mainly used for data organization and management. To revise intermediary and final modelling results the Mesh Viewer tool\(^2\) was used to display 3D output visualisations that were created by applying functions from the GTS library.

Because of the experienced need to differentiate between the stockpile and the bunker area a data structure was introduced that provides a com-

\(^1\)http://gts.sourceforge.net/
\(^2\)http://mview.sourceforge.net/
mon access object to parts of the surface by different plugins. The surface descriptor is the container object for three surface objects, the custom surface, stockpile surface and the bunker surface and it is depicted in figure 5.2.

The surface objects themselves comprise of a GtsSurface as can be found in the library mentioned above, a list of vertices belonging to the surface, a bounding box determining the metric extent of the surface and a name to reference the object. The GtsSurface object contains bounding box and vertex structures itself but it imposes that a valid triangulation (not necessarily delaunay) for the surface must be available to be fully deployable. In cases where a triangulation is not strictly necessary, like for the matrix model plugin, the GtsSurface is left empty and only the other fields of the surface structure are considered.

![Figure 5.2: Surface descriptor data structure.](image)

Every surface is stored within a surface descriptor object so they can easily be compared or merged into a single object representation. Therefore after the conversion to the internal format is finished a surface descriptor object for every point cloud as well as the base surface is available and they are summarized to a surface descriptor set. The custom surface of a surface descriptor is used to provide visual output for the stockpile area as a whole whereas the stockpile and bunker surfaces are used to generate the corresponding volume measurements.

## 5.3 Model Module Operation

The functionality of the model module is determined by plugin handlers that execute corresponding plugins where model acquisition, creation and analysis tasks are implemented. The model engine manages references to
CHAPTER 5. MODEL MODULE

the needed handlers where installed plugins are processed.

To revise the state of the module from the outside, a request to the model server is sent that is forwarded to the model engine, which delivers an answer dependant if it is in idle, running or error mode. Contrary to the sensor module, where all sensors run simultaneously and independent of each other, the functionality of the model module is executed sequentially. Therefore it is sufficient to return the status of the actually executing task. Additionally a progress indicator can be included in the user interface initiating modelling, which is updated to report the advance of model computation to the user.

The following description of the model module functionality is based on the typical workflow the module must satisfy to correctly interpret data from different sensors to retrieve volume measurements of the observed area. A representation of this workflow can be found in figure 5.3 where the different handlers are applied from top to bottom.

For our purposes three different handler objects were identified that are applied in succession to retrieve evaluable 3D models. The sequence of handlers is determined over the plugin handler entry within the configuration file and if a certain sequence among plugins under a single handler is required the plugins can be named in an alphanumeric style.

First the input handler is responsible for acquiring data from different sources and preparing it in a way to be usable by following stages. Depending on the type of input, the transformation handler prepares sensor data to

![Figure 5.3: Model Data Flow.](image-url)
be comparable with each other and the base surface to forward it to the last stage. Finally, the model handler manages the assembly of the three dimensional model and the volume measurement that gets derived from it. After modelling and volume extraction is completed, the model engine delivers measurement reports to a server hosting measurement results and informs the user about successful operation. Data exchange between handlers and plugins is organized over data structures referenced by the model engine. In the following sections the details of every plugin are described.

5.4 Input Handler

The input handler is responsible to collect and preprocess data from the sensor module as well as the base surface object that is needed for the following modelling steps. For our purposes two plugins are defined that deal with this task.

5.4.1 Sensor Surface Plugin

The sensor input plugin gets passed the identification of a particular scan instance whose data is requested from the sensor server or locally, dependent on the distribution of system components. As already mentioned, a scan instance contains a meta configuration file describing different high level parameters of the sensor devices as well as files containing the raw data from the recordings of every single sensor. The sensor module low level properties, like sensor connection port or device internal settings, are only interesting to the sensor module and not forwarded to the model module. Other parameters like transformation properties needed for 3D modelling are also contained in the meta configuration file. They are only used by the model module and are unneeded by the sensor module. They are included in the meta configuration to decrease the amount of static configuration for the model module so it can be used for arbitrary sensor acquisitions. This way only the location of the base surface and the bunker properties need to be explicitly configured for the model module. For our initial installations with two laserscanners and a distance measurement sensor, one data file for every involved laserscanner and one file for the position measurement for a particular scan instance are retrieved.
5.4.2 Base Surface Plugin

The base surface that represents the ground surface of our modelling problem, is acquired by the base surface plugin. It can be requested from the sensor module or as in our case can be made available from local storage. It is accessible in the AutoCAD dxf file format and the libg3d library is used to read the data contained within the file and to store it directly in the format used by the model module.

It is provided from a terrain measurement using the classical approach where the most prominent vertices of an area are recorded and meshed up to describe the base surface model. Because the dxf file format already contains a triangulated surface mesh of the base area it is not necessary to apply any conversions on the data and therefore the base surface can be directly stored in a surface descriptor object which makes the application of the transformation handler plugins on the base surface unnecessary. This is shown in the model data flow image and it can be seen that the results of the base surface input plugin are directly passed to the modelling handler in contrast to the laser scanner surfaces that first need to be converted by the plugins of the transformation handler.

5.5 Transformation Handler

After raw sensor data is received from the sensor module, it needs to be converted to the internal format of the model module that consists of 3D surface meshes that afterwards can be analysed with the algorithms applied by the last handler. The plugins of the transformation handler have both a data format conversion and a surface complexity reduction functionality.

5.5.1 Sensor Preprocess Plugin

First, laser scanner data that is given as a successive set of two dimensional scan slices, which are encoded as a list of point objects in polar coordinates are transformed to equivalent 2D cartesian coordinates. The start and end timestamps of every scan slice measurement are used to calculate an interpolated timestamp for every vertex within a scan slice.

3http://gna.org/projects/libg3d/
For the measurement of the distance sensor a linear time distance relation between single measurement values and the corresponding timestamps can be established. To find the third coordinate for any vertex within a slice set, the timestamp of the vertex is used to extract the corresponding time value from the distance measurement. By using linear interpolation of time and distance values the interpolated distance value for a single laser scanner vertex can be found from the distance measurement. Applying this process to all vertices within a slice set results in a single point cloud given in the local coordinate system of a particular laser scanner, while the distance data set is no longer needed and can be disregarded.

Because of the indeterministic runtime behavior of single sensors it may be possible that the laser scanner devices already started with their work whereas the distance measurement is still launching the running state. As a result laser scanner data will be acquired for where no distance measurement is available. These slices also need to be deleted by the plugin and they only appear at the begin and the end of a laser scanner data set.

5.5.2 3D Conversion Plugin

To obtain a global 3D model valid for all different surfaces the surface vertices from the sensor devices given in local coordinate systems have to be transformed to the common reference coordinate system of the base surface. The transformation parameters are provided over the configuration information of the scan instance and relate to the origin chosen within the base surface. In the course of applying the local sensor transformations a surface descriptor object is instantiated for every scanner surface and all the vertices of a slice set are written in the converted form to the surface object describing the surface as a whole (called custom surface; see below at bunker plugin section).

After the 3D transformations were applied, surface objects from laser scanners and the base surface are merged by simply overlaying the point clouds which is a valid operation as all surfaces are already given within a single common coordinate system. This can also be done in the modelling stage and filters of the next handler will be applied to every surface individually.

Especially for 3D visualization the dense point cloud received from the sensor devices is reduced to a number of vertices that makes visualization
processing possible in a practical time.

After the 3D Conversion Plugin has completed all surfaces are available in the same coordinate system and can therefore be treated in an equal fashion. For the following bunker and matrix plugins it is sufficient to consider all surfaces in succession and apply the corresponding algorithms on them whereas for the geometric plugin an explicit join of sensor surfaces must be generated (resulting in a new compound surface object) because a triangulated surface mesh of all points must be available before applying geometric modelling.

5.6 Model Handler

The model handler manages plugins that are dedicated to build up and evaluate the 3D model that is by now available in an internal format. Actually three different plugins are defined where each has its own properties and is used to produce volume measurement output regarding to user requirements. The bunker and matrix model plugins are used in the actual deployments of the system whereas the geometric plugin is used for testing purposes, mainly to generate visual output for easy plausibility testing.

5.6.1 Bunker Plugin

After initial tests it was quickly seen that modelling and volume evaluation of the bunker structures with one of the approaches that are used for the heap structures lying on the main, more homogeniously formed part of the stockpile did not lead to the desired results (also due to difficulties regarding the mapping of scanned bunker surface to base bunker surface). To obtain quick and comparable output, another plugin was defined that evaluates the bunker part of the stockpile in an alternate way that automates the process that was used up until now but needed to be done manually. It is described in the following paragraphs.

The bunker model plugin achieves two main tasks. First it provides a cut operation to separate the main stockpile surface from the bunker area and the second task is to evaluate the volume of the bunkers. These two parts could have been divided over two separate plugins but was combined in a single plugin because in the actual version of the system there was no need to have one of the two functions available separately.
5.6.1.1 Surface Splitting

The bunker surface can be clearly distinguished from the main stockpile area as both are located at different adjoining areas. First the base surface will get splitted, as the result of this operation will be needed for the separation of the scanner surfaces.

In the top view orientation a separation line is layed right to the bunker part of the base surface in a way, so that all triangles of the bunkers are fully located at the left side of the line but for triangles belonging to the stockpile surface at least a single vertex lies right to the separation line resulting in the triangle being assigned to the stockpile surface. Graphical examples of the different surface mesh objects in top view representation are depicted in figure 5.4 a) - c) where only the starting part of the area is shown to keep different structures recognizable.

After the bunker surface has been determined the bounding box of the bunker is calculated and the maximum of the bounding box in the $x$ direction serves as the separation line value for dividing the vertex clouds originating from the scanner surface. Because the scanner surfaces are not available in a triangulated form at this stage of modelling, the above routine which investigates the positions of single vertices of a triangle does not work. Instead it has to be determined for every vertex separately, if it belongs to the bunker or the stockpile surface. And therefore the maximum of the base surface bounding box in the $x$ direction can be used as a reliable parameter to clearly distinguish the bunker and stockpile surface of a scanned point cloud.
a) Base custom surface including separation line.  
b) Separated bunker surface.  
c) Separated stockpile surface.

Figure 5.4: Surface bunker separation properties and output.

5.6.1.2 Bunker volume extraction

After the surface objects are available in their separated form the volume of material contained within the bunkers gets evaluated. For this purpose the height of material contained within a bunker gets extracted and this is done following the subdivision of the measured area according to the assigned material ranges. This approach is chosen so it is possible to associate the volume of a certain material with the same material on the stockpile surface to receive the overall volume for the specific material.

For every material a bunker fill level array is established that corresponds to the range for which a specific material is set. The fill level array is a sequence of fill level items alongside a part of the bunker surface where every item gets assigned the minimum height at the part of the stockpile where the item is located. A fill level item describes the minimum height of all vertices that fall within the fill level indicator box that is located near the rightmost end of the bunker surface seen in a front view perspective. Figure 5.5a) shows a front view schematic of a bunker where an example of a single scan slice and the corresponding fill level indicator box are depicted. In figure 5.5b) a top view of a part of the bunker surface is seen, overlaid with the corresponding fill level array to see at which position the array is established. Figure 5.5c) shows a side view plot of a part of the bunker surface, where this is a simple plot of all the heights contained within the fill level array where only a single material range is specified. A heap structure contained within
the bunker can be clearly seen on the left side as well as corresponding bunker walls that are represented by the bars within the plot. A 7-bin median filter was applied as artefacts from the area above the bunkers produced many unwanted outliers that are smoothed this way.

Figure 5.5: Bunker fill level diagrams.

Figure 5.6: Bunker content diagram for a single bunker.

After the fill level array has been established the volume for set material ranges within the bunkers need to be determined. Therefore a manually determined mapping that can be found in figure 5.6 is used that provides
an approximation for the volume contained within a bunker depending on the height any material reaches. It assumes an equal distribution of bunker widths and has to be divided by the number of fill level items that are contained within a certain bunker. Afterwards the volume represented by a specific fill level item can easily be extracted according to the modified mapping. The bunker volume corresponding to the material range under consideration needs to be added to the stockpile volume for this range to yield the overall volume for the considered material range of the stockpile.

5.6.2 Matrix Plugin

The matrix model plugin is the part of the system where a measurement along main stockpile area is evaluated and whose volume calculation results are later presented within the according user interfaces. It makes use of a modified modelling approach that is actually applied by most industrially available products due to its simplicity and sufficient accuracy of results.

The matrix model plugin evaluates the stockpile part of the base and the scanned surfaces. A matrix surface is a compact representation of a surface area, and two different matrix surfaces are required, where the first is a mapping of the base surface to a matrix model named base surface matrix and the second describes a matrix model for the measured surfaces known as object surface matrix.

The matrix model of a surface is a subdivision of the surface into a two dimensional grid where all cells are equally sized and the matrix covers the whole surface seen from a top view in $x$-$y$ direction. Single cells of the matrix get filled with a single vertex positioned in the center of the cell that represents the average height of all the vertices of the original surfaces that fall within this cell. Every cell in the matrix is equally sized and the size of the cell is a tradeoff between accuracy and calculation speed and an evaluation is given in the last chapter. By the subdivision of a surface into a regular grid, implicit filtering takes place that reduces the amount of considered vertices depending on the chosen cell size. The usage of vertex structures for representants of the cell heights is convenient because they can later be used to establish a surface triangulation that can easily be visualized. Example visualizations are given in figure 5.7 where the first three images show the same heap structure using the same perspective but with different display options. Figure 5.7d) is a top view of the scan area to depict the
regular mesh that is created by applying the matrix plugin.

![Figure 5.7: Example output generated by the matrix plugin.](image)

Two such grid models have to be established, one for the base surface and another one for the scanned vertex clouds and they are called the height field of the matrix model. As the surfaces from different laserscanners are already transformed into a common coordinate space the vertices of all laserscanner surfaces are combined to determine the average height of a particular cell.

The base surface, that has a rather sparse vertex population, needs to be treated in a special way because for many cells within the matrix no direct height can be calculated as no representative vertices are available for this cell. Therefore, first a matrix with the $x$, $y$ dimensions of the base surface is created whose height value is zero for all cells. Then this matrix is projected onto the base surface, which yields a base surface matrix where the height values are set with the real height values as given by the base surface.

To retrieve the volume of the considered area from the matrix model in the last step both surface matrices get inspected in an $x$-$y$ direction from
a top view and for all places where the object surface matrix and the base surface matrix cells are filled the volume of the enclosed rectangular solid, determined by the two height field values and the cell size, is calculated. The single volumes are accumulated to yield the overall volume of the object measured.

The setting of material ranges is used to differentiate between the volumes of different heap objects. For the final evaluation result, the bunker volumes retrieved in the previous plugin need to be combined with the volumes of the heap objects from the stockpile to give the overall volume for the area under consideration. Fortunately, the bunker fill levels and volumes are already set up in accordance to the corresponding material ranges and therefore it is only necessary to add the bunker volume of a certain position within the bunker surface to the appropriate position within the stockpile surface.

5.6.3 Geometric Plugin

The geometric model plugin is a side product of the main system and was developed initially to have reliable surface representations that can easily be used to evaluate model plausibility over their visualizations. The routines to output graphical 3D meshes used throughout all stages of the modelling process were also developed in the course of implementing this plugin.

An approach for how volume measurement can be achieved is sketched in the following paragraphs. The considered visualizations generated by this plugin on the other side intuitively seem to produce correct graphical output that is used to initially revise the possible quality of any measurements.

As with the matrix plugin, again two different surfaces are required that will be compared to each other to retrieve volume measurements. Within the geometric approach, surface objects must be available in a form of a triangulated mesh because the surfaces are analyzed while keeping their geometric properties. The base surface is already existing in a triangulated form and can therefore be used in an unaltered fashion. The laserscanner surfaces on the other side are only available as a point cloud, localized relatively to the base surface without direct relation between each other. Therefore the laserscanner surfaces first have to undergo an explicit join operation which results in a single combined object surface structure. The join operation simply creates a point cloud that consists of all the points originating from several sensor devices.
Afterwards, the object point cloud needs to get triangulated to obtain a closed surface object. For this purpose, the delaunay triangulation that is contained in the GTS library is applied as it behaves optimally regarding minimality in area of produced triangles. To build the meshed 3D surface, first only the $x$ and $y$ coordinates of every vertex are used to establish the corresponding delaunay triangulation that was originally defined for two dimensional point clouds. To obtain a 3D model, the $z$ coordinate of every vertex is considered again in following calculations. There is also the possibility to perform 3D triangulation (called tetrahedralization, cf. [7]), but this causes higher computation costs and was not needed for our purposes.

Because of the common coordinate space the vertices of the object surface can be projected onto the base surface. The projection is achieved by building a line with a second imaginary vertex that lies in the opposite $z$ location, beneath the base surface. This line can be used to calculate the point where the line intersects with the base surface area where the intersection point represents the projection of the considered vertex onto the base surface. The resulting point cloud of intersection vertices can be interpreted as the same triangular surface mesh as the object surface this time located on the base surface and it is called the \textit{intersection surface}.

The 3D space between triangles of the object surface and the intersection surface virtually form little prisms whose volume can easily be determined. By accumulating these partial volumes the overall volume of a material heap or a sequence of heaps is calculated. At the moment the differentiation of heap objects by the setting of material ranges is not considered by the geometric plugin and therefore the observed area is always treated as a single surface.
Figure 5.8: Example output generated by the geometric plugin. For the surface shown in d) - f) a filter of $n = 16$ was used.

As for the previous plugin a couple of visualizations are presented in figure 5.8 a filtered and an unfiltered version of the same heap structure is shown. The applied filter considers for a parameter $n$ only every $n$th vertex within a slice meaning that the amount of vertices within a surface is decreased by a
factor $n$. For quick visual evaluations the filtered version is mostly sufficient as the data and calculation requirements increase by the factor filtering is decreased.

Figure 5.9: View of stockpile with base and object surface combined.

For the sake of completeness a visualization is shown in figure 5.9 that depicts the stockpile surface of the base surface and the object surface in a single view.
Chapter 6

End User Interfaces

The user interfaces present visual summaries of the data processes underneath and are used to control the operation of both modules. According to the requirements of the actual implementation two specific user interfaces were realized that can easily be replaced if different needs arise, due to the generic system architecture approach.

The crane UI is the part of the system where all the subcomponents are triggered and controlled in an integrated way. The user is able to start and stop single scan processes and will obtain different output during and after the scan. The interface dedicated to the dispatcher employee (called dispatch UI) gives a condensed overview of the model alone with unneeded details left behind.

6.1 Crane Operator User Interface

The crane operator is the person who is responsible for data acquisition so the model module can afterwards evaluate the resulting model and extract the necessary values. For this purpose a user interface was developed where a screenshot of the GUI in idle mode can be found in figure 6.1 where the results of a previous run are still displayed. Due to the human component in form of the crane operator some constraints need to be obeyed for successful recording of data that is usable for modelling afterwards. At the moment mishandling of the system is covered only to a certain extent and special

\footnote{Many thanks to Simon Vogl (project supervisor), Simon Opelt and Philip Aumayer who provided multiple support in implementing and testing the actual prototype.}
situations due to user misbehavior are outside the scope of this work.

To initiate a scan instance, the crane operator starts the system by pressing the start button and begins to move the crane vehicle alongside the heap objects that need to be observed. During a scan instance it is possible any time to change the material identification, especially when the material changes at the next heap object. When the interesting area has been observed the user presses the stop button and the system starts the modelling process as described above. Modelling results of the latest observed scan instance can be found on the right side below the distance measurement widget. Here a list of material heap objects and their volume is given in the sequence they were scanned (the first scanned heap is the topmost element in the list). This list is updated whenever a scan instance has been recorded and the following model stage finished. This way the crane operator can do a subjective plausibility test. If the results are completely misleading the scan may be redone. Any postconfiguration of the model instance as may be needed by remodelling, can not be altered by the crane operator.

The two bigger displays on the upper left side show a life picture of the actually scanned profile during a running scan instance. They give a hint when a object heap has ended and the material ID may be changed. Under the image label on the right side, the actual position, in meters, from the origin at the end of the area is shown. This data is provided by the different sensor entities of the sensor module. The provision of data for the online views is not included in the system architecture above as this is outside the main system workflow. The online views are maintained by the single sensor entities that are served in idle times within low priority threads when the main data acquisition thread of a sensor is inactive.

The sensor module is implemented in a way to signal the user that all sensors are fully running when the first feedback visualizations arrive at the crane UI. This way the physical pollution of the sensor devices mentioned above can easily be read off the actual profiles displayed on screen. If no clear surface can be detected but a cluster of points is drawn in the top center of the corresponding widgets it is a strong indicator that the field of view of the laserscanner needs to be physically cleaned.
CHAPTER 6. END USER INTERFACES

6.2 Dispatcher User Interface

The dispatch user interface on the other side presents a more detailed and summarized view about the created model and extracted data and allows the adjustment of parameters afterwards. A screenshot of the dispatch user interface can be found in figure 6.2.

![Dispatcher User Interface](image)

Figure 6.2: Dispatcher User Interface.

Here a side view profile of the stockpile area is presented to the user that represents a summary of the calculated model instance. The 3D outputs used for evaluation of different modeling steps are actually not used as additional visualizations for the user. This is an issue for further work.
This approach was also chosen due to a practical need because it may be possible that the crane operator entered wrong values while recording the corresponding scan instance, but it may still be usable. In the dispatch UI the lengthside profile depicts every heap object contained in this scan instance, the borders of the object (in real valued physical dimensions), the material the heap consists of and the volume of every contained heap. If for example the crane operator set the borders of the heap wrongly, the dispatching user has the possibility to change these borders and initiate a recalculation of the volume extraction. If the dispatcher is satisfied with the results of the model instance, various reports can be created in the style as needed by the bookkeeping department.
Chapter 7

Conclusions

As can be seen from the above chapters, the specification and implementation of a multi sensor stockpile measurement system is a complex task and a lot of other approaches are possible that may be of concern to further research. The way the desired functionality was implemented represents a compromise between customer requirements, feasibility of system components and best practices within current research. Therefore some issues are not treated to the extent they may have deserved but a whole system specification is provided that involves all important topics regarding the volumetric measurement of large scale outdoor stockpile objects and give hints for extensions in various ways.

7.1 System Performance

The performance of the system is mainly dependent on the behavior of the model module and therefore this part focuses on the runtime performance of this module specifically the matrix plugin.

Before an actual runtime evaluation is accomplished the example scan instance taken to provide the base for this evaluation needs to be explained. A reference heap was piled up whose value was roughly known against which the accuracy of the model module was tested. The corresponding measurement created a scan range of 19.08 meters and an amount of around 200,000 vertex points for each of the two applied laserscanners. The exact number differs between laserscanners due to different configured scan angles and indeterministic runtime behavior. The number of recorded surface points is
also dependent on the movement speed of the crane vehicle during the measurement. A fast and a slow movement speed are available where the fast speed needs one fourth of the time to move the crane from one point to the other compared to the slow speed. For our evaluation example the slow speed was used whereas when using the fast speed a point cloud of 50,000 points would have been recorded by a single laserscanner.

Another point is that in this example only a small part of the stockpile is considered while measurement of the whole stockpile area at once can create considerable amount of data. For example an overall scan of the stockpile area used in this description created a scan instance that required around 90 MB of disk space where the crane was moved using the fast speed. The same scan instance would have needed four times this space if the area would have been measured using the slow speed.

The laserscanner and GPS device are configured to record data with the same frequency (10 Hz) but where a laserscanner records a series of angles and corresponding distances the GPS device only produces a single location value. Therefore the amount of data produced by the GPS device is only a small fraction of the data amount retrieved from the laserscanner and lies around 0.25 %. So the main bulk of data originates from the laserscanner devices and the overhead introduced by the GPS device and the meta configuration file is marginal. The scan instance that was used for performance analysis and also throughout the visualizations within this document had a size of 9.452 MB.

The results of matrix model performance evaluation are summarized in table 7.1 where the cell size of the model matrix is varied and the implications on running time and volume measurement accuracy are recorded. The dimensions of the base surface in $x, y$ direction are around 49 and 415 meters respectively. For sake of brevity here only rounded values are presented. For evaluation purposes a reference heap that had a volume of around $410 \text{ m}^3$ was prepared that had a physical length of 19 m. As we deal with compressible bulk material the exact volume value cannot be definitely determined anytime and the final volume values are determined by the dispatcher by applying constant weight factors for different materials.
CHAPTER 7. CONCLUSIONS

<table>
<thead>
<tr>
<th>cell size [m]</th>
<th>matrix size</th>
<th>Time [s]</th>
<th>Volume [m³]</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5 x 42</td>
<td>0.52</td>
<td>486.96</td>
</tr>
<tr>
<td>5</td>
<td>10 x 83</td>
<td>0.59</td>
<td>437.90</td>
</tr>
<tr>
<td>2</td>
<td>25 x 208</td>
<td>1.26</td>
<td>419.83</td>
</tr>
<tr>
<td>1</td>
<td>49 x 415</td>
<td>4.62</td>
<td>417.62</td>
</tr>
<tr>
<td>0.5</td>
<td>97 x 830</td>
<td>20.73</td>
<td>385.96</td>
</tr>
<tr>
<td>0.1</td>
<td>482 x 4146</td>
<td>2005.83</td>
<td>478.31</td>
</tr>
</tbody>
</table>

Table 7.1: Matrix plugin performance evaluation.

From the table it can be seen, that the runtime increases considerably with decreasing size of the matrix cells yielding bigger matrices. The best result was retrieved when a cell size of around 1 m was chosen. It is clear that with increasing cell size the volume result will get biased due to a lower resolution where too much information gets lost.

When switching to a higher resolution it can be seen that accuracy begins to move away from the target result again. This has several possible reasons. One is that while the crane vehicle is in movement the whole construction is more or less shaking and so do the laserscanner devices. Therefore little artefacts appear in the recording dependent on how much interruption from a smooth movement was experienced. Another reason is that transformation parameters to build up the 3D model are not exactly correct which also introduces a source of error. In our first prototype installations these error sources have been dimmed to produce as less impact as possible and the system has reached a state to deliver results within the required accuracy.

7.2 Summary and Further Work

In the last paragraphs a few options are depicted that give an outline for further work that can be done to optimize the system as a whole and parts of it.

System features that were partially implemented and not mentioned in this work because they have no impact on general system workflow are a configuration interface to set up sensor properties over the crane UI or dynamic addition and removal of sensor devices and resetting system configuration during system runtime.

An issue that is of concern in the near future regards the difference be-
between the idealized base surface and the real world surface without any material on it. It has shown that the base model represents parts of the stockpile area very well whereas in certain locations strong deviations can be found which minders the quality of measurement results. An approach to handle this problem is to achieve zero measurements where a part of the stockpile that contains no material is observed. Unfortunately it is not possible to achieve this for the whole stockpile at once as is logistically impossible empty the whole area. Single zero measurements of parts of the stockpile can be used to update the considered parts within the base model surface.

More work can be done within automatic heap object segmentation or error detection and correction. Another interesting question is how the system can be adapted to serve for acquisition and evaluation tasks within other domains.
Bibliography


[19] Raphaël Labrade, Cyril Royere, Dominique Gruyer, and Didier Aubert. Cooperative Fusion for Multi-Obstacles Detection With Use of Stereovi-


