FEATURE EXTRACTION WORKFLOWS FOR URBAN MOBILE-
TERRESTRIAL LIDAR DATA

by

Gregory John McQuat

A thesis submitted to the Department of Geography
In conformity with the requirements for
the degree of Master’s of Science

Queen’s University
Kingston, Ontario, Canada
(May 2011)

Copyright © Gregory John McQuat, 2011
Abstract

Mobile Terrestrial LiDAR (MTL) is an active remote sensing technology that uses laser-based ranging and global positioning systems (GPS) to record 3D point location measurements on surfaces within and near transportation corridors, such as along a railroad track or a street. This thesis examines geovisualization for improving user-oriented workflows and also examines geographic object-based image analysis (GEOBIA) for the development of automated feature extraction. A LiDAR sensor-centric perspective during the data acquisition phase is used to organize data for the user and to transform the data into a 2D reference frame for object-oriented image analysis of MTL data.

Organizing the display of MTL data relative to the scanner presented new opportunities for visualization techniques and was an effective method for communicating space that was scanned, or not, in an urban scene. It offers new avenues for quality assessment of MTL survey of urban environments by explicitly displaying gaps in data coverage. A number of techniques for navigating and visualizing data from a sensor-perspective are examined.

A novel sensor-perspective transformation of MTL data from three to two dimensions enables analysis of MTL data in common GIS and image-processing environments. GEOBIA software (Definiens’ eCognition) is used to construct a procedural feature extraction workflow. The procedures are constructed with semantic classes, data processing rules and functions that drive geometric segmentation and feature recognition. Geometric regularities in urban scenes and knowledge about spatial and semantic relationships are incorporated into the rule set. The results are fluidly integrated back into a GIS environment.

Investigation of alternative approaches to handling MTL data such as those carried out in this thesis are essential if this technology is to see widespread use.
Acknowledgements

First and foremost, I would like to thank my thesis advisors, Robin Harrap and Dr. Paul Treitz, for giving me the opportunity to pursue this work and for their invaluable guidance and support. I am grateful to Robin Harrap for encouraging me to continue my education at the graduate level and am especially grateful for his sincerity in all matters since we first met five years ago. I am grateful to Dr. Paul Treitz for sparking my interest in remote sensing, for his encouragement and for an intellectual environment where I felt free to explore and to learn.

I would like to thank Craig Sherriff and Kresimir Kusevic of Terrapoint Inc. for their help and invaluable feedback during our meetings in Ottawa and Kingston. I extend special gratitude to Paul Mrstik of Terrapoint Inc. for his guidance and companionship as our paths frequently crossed. I would also like to thank Michael Greenspan of Queen’s University Department of Electrical and Computer Engineering for his essential feedback on aspects of this project.

LiDAR technology is expensive and Mobile Terrestrial LiDAR technologies are innovative and not readily available for academic research. This research was supported by research grants through Ontario Centres of Excellence (Development of Feature Extraction Tools for Mobile Terrestrial LiDAR), GEOIDE (GeoEduc3d Project: Mobile Games for Education) and by a research grant to Paul Treitz from the Natural Sciences and Engineering Research Council of Canada.

To Mike Martin, Karin van Ewijk, and Maarit Middleton, many of the ideas and methods in this thesis arose out of the countless conversations that we had. Thank you for the discussions and the companionship, I am indebted to you all.

Finally, I want to extend my heartfelt gratitude to my partner, Heather McQuat, for her inexhaustible support and understanding as we navigate this life. For Duncan and William.
# Table of Contents

Abstract ............................................................................................................................ ii

Acknowledgements ........................................................................................................... iii

Chapter 1: Introduction .................................................................................................... 1
  1.1 Introduction ........................................................................................................ 1
  1.2 Mapping Cities ................................................................................................. 1
  1.3 Positioning Technologies ................................................................................ 2
  1.4 From Data to Information ............................................................................. 3
  1.5 Complexity ....................................................................................................... 4
  1.6 Geographic Information .............................................................................. 6
  1.7 Levels of Detail ............................................................................................. 7
  1.8 Research Goals .............................................................................................. 9
  1.9 Thesis Outline ............................................................................................... 9

Chapter 2: A Review of LiDAR and MTL Technologies ............................................... 11
  2.1 LiDAR and Supporting Technologies ............................................................ 11
  2.2 Laser Ranging ............................................................................................... 12
  2.3 LiDAR Scanner Components ..................................................................... 13
  2.4 Positioning Technologies ........................................................................... 13
  2.5 Inertial Measurement Unit ......................................................................... 16
  2.6 Data Acquisition System ........................................................................... 16
  2.7 LiDAR Products and Application ............................................................... 17
  2.8 Types of LiDAR Systems ........................................................................... 18
    2.8.1 Static Terrestrial LiDAR ........................................................................ 18
    2.8.2 Airborne LiDAR ................................................................................. 19
    2.8.3 Mobile Terrestrial LiDAR ................................................................. 19
  2.9 LiDAR Processing Software ...................................................................... 21
  2.10 The Mobile Mapping Industry ................................................................. 22
  2.11 The TITAN® MTL System ...................................................................... 23
Chapter 3: Geovisualization Techniques for Mobile-Terrestrial LiDAR: A Sensor Model Perspective

3.1 Introduction
3.2 Feature Extraction
3.3 Geovisualization and Human-Computer Interaction
3.4 Geovisualization and LiDAR

3.4.1 Data Complexity
3.5 Open Space

3.6 A Visualization Tool: MTL Viz

3.6.1 Implementation
3.6.2 Frames of Reference
3.6.3 Modes of Interaction: Location, Scope, and Scale
3.6.4 Visualization Techniques

3.6.4.1 Modelling Scanner Operation
3.6.4.2 Augmentation

3.7 Example Scenario
3.8 Limitations

3.8.1 Reversed Scan Order
3.8.2 Mount Geometry and Pointing Systems

3.9 Conclusions

Chapter 4: An Object-Oriented Approach to Extracting Geographic Information from Urban Mobile-Terrestrial LiDAR Data

4.1 Introduction
4.2 Background

4.2.1 Object-Oriented Image Analysis and Urban Scene Models

4.3 Presented Work
4.4 Methods

4.4.1 Study Area and Data
4.4.2 Data
Appendix A MTL Viz – Processing® Code ................................................................................ 119
Appendix B Class Descriptions ................................................................................................. 134
Appendix C Customized Metrics .............................................................................................. 137
Appendix D Main Procedure .................................................................................................... 139
Appendix E Sign Recognizer: Procedural Workflow Diagram .................................................. 144
List of Figures

Figure 2-1. Example of a typical urban street corner point cloud collected using the TITAN® MTL system in Kingston, Ontario, Canada ......................................................... 12

Figure 2-2. A mobile system uses accurate positioning technologies. The internal measurement unit (IMU) fills in the gaps between the GPS epochs and provides pose. Timing is the key to merging data from various sources ................................................................. 14

Figure 2-3. Differential GPS positioning. Anomalous variations in base station positions are used to correct the position of the mobile system ............................................. 15

Figure 2-4. IMU Error Amplification. The IMU keeps track of small changes in position (X, Y, Z) and attitude (pitch, yaw, roll) of the system. Accuracy is important because small errors in angle measurement are exaggerated over distance such that pose errors have much higher impact than position errors ............................................................... 16

Figure 2-5. Example schematic of mobile system architecture and data flow. (Adapted from Glennie 2009) ................................................................................................. 17

Figure 2-6. Data fusion of Leica HDS6000 static scanner data to the TITAN(R) point cloud. The area shown is at the intersection of X and King St. in downtown Kingston, ON. Note the high resolution Whig Standard building and the lower resolution MTL data for adjacent buildings providing the context in downtown Kingston, ON, Canada (Image courtesy of Matt Lato, Queen’s University) ........................................................................ 21

Figure 2-7. TITAN® and acquisition vehicle. The TITAN® pod is mounted atop an adjustable lift and data cables run into the vehicle cabin where the crew monitors and controls the DAS. Here TITAN® is shown installed on a truck equipped to drive on railways. (Image courtesy of Craig Sheriff, Ambercore Inc.) ................................................................. 24

Figure 2-8. TITAN® operational environment. Note that rooftops will be absent while the roadway and façades are captured with substantial detail ............................................. 25

Figure 3-1. Existing visualization perspectives for a TITAN® LiDAR point cloud of upper Princess Street in downtown Kingston, Ontario, Canada. Top and Centre: 3D virtual environment (Merrick MARS) perspectives looking down the sidewalk and a bird’s eye
view from above the scene. The curve shows the acquisition trajectory. **Bottom:** Top-down view of MTL data in a GIS environment

Figure 3-2. Point density and point patterns are a function of distance and movement of the sensor. The point density diminishes with distance and point patterns vary on a wall that is nearly parallel to the scanning plane. While the arrow indicates the sensor perspective of the house, the sensors’ position is not intuitively obvious from the points alone. 

Figure 3-3. Visualizing Open Space. Arrows indicates the sensor perspective on the data.

Figure 3-4. Along-track and across-track perspectives of the same LiDAR points as seen from the LiDAR scanner, relative to the data acquisition trajectory.

Figure 3-5. Various metrics used for the sensor frame of reference. The image shows the TITAN(R) system mounted on a pickup truck and the relationship between the beam path and optional metrics.

Figure 3-6. Top view of width-adjustable data slice. A user moves along the trajectory to navigate the scene and sets the width of the data slice drawn to the screen. The slice is orthogonal to the view shown here.

Figure 3-7. Point placement relative to the current focus. Different colouring or shading is applied to point behind and in front of the current position to communicate directionality of the scan.

Figure 3-8. Trailing, semi-transparent triangles show directionality of acquisition and communicate scanned space in the scene.

Figure 3-9. Open space polygons constructed from the point set and knowledge of scanner-specific robotics.

Figure 3-10. Example workflow with MTL Viz.

Figure 3-11. The effects on data distribution when the MTL system is travelling in an arc. The order of returns traces backwards across the building wall to the inside of the arc and will be retraced as the system continues moving down the street.

Figure 3-12. TITAN(R)® sensor offset between side and rear-facing scanners. Top View.

4-1. Image-Object and Urban-Class Hierarchies.
Figure 4-2. Definiens’ eCognition® user interface. Viewports display various views showing classification or image-object metrics that are in turn used to develop the rules and procedures that make up the process tree. 63

Figure 4-3. Object-oriented workflow. An initial segmentation into image-objects is followed by many iterative steps of image-objects modification and classification before exporting usable geographic information. 65

Figure 4-4. Top: Data is represented in height from scanner (left) and height from Geoid (right) for two different sections along Princess Street in Kingston, ON. Note the data gaps in the left image caused by occlusion. Bottom: Data is along the vertical axis is displayed by angle relative to the scanner. Occlusion caused by foreground objects is virtually eliminated when considering the scanners perspective. 68

Figure 4-5. Study Area and Data. 70

Figure 4-6. Left: TITAN® mobile mapping system mounted on a pickup truck. The adjustable boom is shown in a lowered position. Right: A schematic view of data collection seen along the system trajectory. (Photo courtesy of Craig Sherriff, Terrapoint Inc.) 71

Figure 4-7. Urban scene feature extraction workflow. 73

Figure 4-8. Information derived from the scanner-point coordinate pairs: Height From Sensor (HFS), 2D distance (2D Range) from sensor and the angle from level at the sensor height (theta). 74

Figure 4-9. The axes are scaled for interpolation. Top Left: Scan lines shift to the right and have large gaps from one scan line to the next because they were acquired from a moving vehicle. Top Right: Matching precision and resolution of the non-isometric axes gives a more uniform point distribution suitable for interpolation. Bottom: Points along approximately 20m of trajectory in scan coordinates. Note the gaps in point coverage. 76

Figure 4-10. TIN data structure (left) and extracted analysis mask (right). Size of TIN triangles is used as a proxy for large gaps in point coverage. 77
Figure 4-11. Interpolated raster images. Intensity (top left) and various types of spatial information such as range (top right) are interpolated. Left: UTM Easting, Northing, and Height contours are shown.

Figure 4-12. The Sensor-Perspective Scene Model. Schematic indicates key features where the height of the ground and distance to facade relative to the scanner are important.

Figure 4-13. Adjusting for the forward look angle. Top-down view. Note: 10° line angle is exaggerated for illustrative purposes.

Figure 4-14. Expected surface orientations from TITAN® scan on an urban scene. This image shows the results of the first feature recognition stage that separates the scene into image-objects with relatively smooth surfaces.

Figure 4-15. Large surface-growing process. Flat, Front, and Side surfaces are considered independently. Some image-objects are classified into ground, façade, and sidewall.

Figure 4-16. Second derivative of mean difference to neighbours. Greater rate of change (brighter) signifies sudden change in surface orientation while low rate of change indicates a smoother surface.

Figure 4-17. Sign recognizer. A traffic sign typically occupies a region of space at a minimum distance from scanner, is expected to be found within a certain height range above the ground, and is associated with high intensity returns.

Figure 4-18. Step and Curb recognizers. While similar in geometry, relative proportion to ground differentiates steps from curbs.

Figure 4-19. Automobile recognizer. Minimum and maximum height from ground and widths of car side panels define the search space. High intensity markers provide additional cues for poorly sampled vehicles or those scanned from one end instead of the side.

Figure 4-20. Example of automobile feature merging. Seed (purple) is merged with adjacent flat and side facing image-objects.

Figure 4-21. Chessboard segmentation of facade image-objects provides localized measures of relative distance.
Figure 4-22. Zones of transition (purple) occur where there is a jump from a foreground object to a surface in the background. Pixels that belong to these zones are identified by a large range difference relative to neighbour image-objects. ................................................................. 91

Figure 4-23. Extracted features from object-oriented analysis represented in a GIS environment. Showing the linear wall features, minimum convex hulls from bay-classified point sets, and the mean location of sign and automobile features overlaid on existing GIS data. .............. 94

Figure 4-24. Comparison of photographs to the scan-image segmentation. The mailbox and disposal bin in the first image set has changed. The awning over the top window in the third set was removed between the time of the scan and when the picture was taken. Vehicles are different as well. ........................................................................................................ 95

Figure 4-25. Perspective views in ArcScene. Height information was used to place the ground (TIN surface) and scene features back into 3D space. Top Left: Front view of façade with bays placed in 3D space. Bottom Right: Enlarged view showing car and street sign placement. ............................................................................................................................ 96

Figure 4-26. Despite the fact that the church does not fit the expected urban geometry along this street scene and the occlusion from large trees on the lot, the recognizers performed well. However, one tree is erroneously classified as building façade, which subsequently led to improperly classified Bay regions. ........................................................................................................ 99

Figure 4-27. Low sampling density of small features makes them unidentifiable. However, a human can still easily recognize the characteristic placement and shape of a parking meter. ..................................................................................................................................... 100
List of Tables

Table 4-1. Distance from preceding position along the trajectory................................................. 75

List of Acronyms

2.5D Two and a Half Dimensional (XY + Z)
2D Two-Dimensional (XY)
3D Three-Dimensional (XYZ)
ATD Along-Track Distance
CCD Charged Coupled Device (imaging array)
CityGML City Geographic Markup Language
DAS Data Acquisition System
DEM Digital Elevation Model
DGPS Differential GPS
GEOBIA Geographic Object-Based Image Analysis
GIS Geographic Information System
GPS Global Positioning System
HFG Height From Ground
HFS Height From Sensor
IMU Inertial Measurement Unit
LiDAR Light Detection and Ranging
LOD Level of Detail
MTL Mobile-Terrestrial LiDAR
TIN Triangulated Irregular Network
TOF Time Of Flight
UTM Universal Transverse Mercator (projection)
WAAS Wide-Area Augmentation System
Chapter 1: Introduction

1.1 Introduction

The development of methodologies for the effective extraction of geographic information from light detection and ranging (LiDAR) data collected with mobile-terrestrial systems is critical to the application of this new technology for mapping cities. Current research is focused on improving user workflows through data visualization techniques and data processing methods that exploit human knowledge and the predictable geometries of urban scenes. This chapter serves to introduce and describe Mobile-Terrestrial LiDAR (MTL) and its current capacity for mapping cities. A brief review of the methods that have been applied to LiDAR scene analysis is presented. Finally, the research goals of this research project are discussed.

1.2 Mapping Cities

In order to manage existing city infrastructure and to plan appropriately for changes and additions to the urban environment, accurate, up-to-date information is required (Dollner et al. 2006). Geographic information systems (GIS) are used to map and manage urban information once collected; advances in positioning technologies have made it possible to efficiently and safely acquire accurate data for mapping purposes at a variety of relevant scales, albeit at high acquisition cost.
Managing city infrastructure requires knowing the exact location and current state of infrastructure such as fire hydrants, hydro poles and roads. Given the dynamic morphology of cities, the requirement for new methods to manage these features extends beyond accurate mapping of location into qualitative assessment of position and the related problem of change detection. For example, a city might be interested in road quality and rate of degradation, the tilt of aging hydro poles, or which homes or businesses have added additional structures in the last five years. The fundamental challenge is, again, how to map and quantify features, detect changes, and to generally facilitate diverse urban operational tasks while staying within realistic budgets. Ultimately, collected data will reside in a GIS database; the question is how to feed that database efficiently.

1.3 Positioning Technologies

GIS and positioning technologies (e.g., Global Positioning Systems (GPS)) have transformed the manner in which we manage and analyze our world. Modern positioning technologies allow accurate data to be collected from mobile systems and thus spatial data can be acquired for larger areas faster and more frequently. This facilitates up-to-date data and broader analyses over time.

There is a driving need to understand and model our environment in almost every sector of society. Civil applications involve managing infrastructure and resources; defense applications focus on accurate mapping and knowledge acquisition; accurate three-dimensional (3D) virtual worlds are used for simulation (Torrens 2007, Batty 1999), education (Daniel & Badard 2010), augmented reality (Harrap & Daniel 2009, Hedley 2008) and urban design (Pritchard 2007, Yang et al. 2007). Capturing data for large areas approaching the size of modern cities is expensive.
Mobile Terrestrial LiDAR is a relatively new system of integrated technologies that provides accurate, high-resolution position (i.e., X, Y, Z) and intensity information from a laser scanner mounted on a vehicle.

MTL is an emerging technology for generating data for detailed 3D datasets to constrain mapping and modeling of urban landscapes. It provides high-quality, high-resolution, georeferenced data out to tens of metres from the vehicle path and so is ideal for use in mapping transportation corridors such as highways and railroad lines (Glennie 2009, Lato 2009, Shan & Toth 2008). The technology meets or exceeds survey accuracy standards while providing unprecedented mapping of corridor infrastructure (Glennie 2009, Mrstik & Kusevic 2009). The real advantages of MTL are: 1) accurate positional data can be collected over a wide area in a small amount of time; and 2) the method removes the need to close transportation corridors during data acquisition, to put people in harm’s way, or to spend large sums of money on traditional surveying. For example, the vehicle can drive with traffic, the operators are within the vehicle, and the system can make tens of thousands of accurate measurements per second. The challenge is in using these measurements effectively.

1.4 From Data to Information

Methods that extract features from airborne and static LiDAR data both take advantage of the particular characteristics of the data and the scan view (Jwa et al. 2009, Hu et al. 2009, Alharthy & Bethel 2003, Clode et al. 2004). However, MTL data is fundamentally different because of being mobile and having a within-scene view. In other words, in a dense urban environment, MTL effectively maps a cylindrical swath through the city, not a birds-eye view
from a distance. As a result, there are issues to overcome within a data processing environment in order to extract features of interest.

Typically, the goals of MTL within an urban setting are to: (i) map urban features at building and block scales; (ii) detect changes to urban features over time; and (iii) detect if large features were added or removed. For example, it is useful to locate roadside infrastructure, to determine if this has degraded over time, or if something is out of place. However, there is currently a lack of methods and tools to extract this type of information from MTL data.

In the first instance, the focus of this research is on mapping; and in the second instance, on different perspectives on visualizing and analyzing the lidar point cloud. Until these aspects are addressed, i.e., the capability of mapping features in the first place is available and efficient, change detection remains elusive. For mapping, new approaches are needed; for example, are there different visualization techniques that might be applied to this problem? In addition, how does the orientation of data representation affect feature extraction processes, especially from the human operator perspective? Further, does mapping the space between point return locations and the sensor position – i.e. the actual path of the returned laser pulse – have value and will approaches involving semantic modelling have important implications to the methods for working with MTL data?

1.5 Complexity

The process of moving from an unorganized point cloud to usable geographic information is non-trivial as there are many challenges:

- objects in the scene move while they are being scanned and the sensor itself is moving;
• sampling density varies with speed of acquisition and surface geometry;
• points rarely fall along object boundaries;
• some materials reflect or refract the LiDAR beam in such a way that there is no
  return pulse to be logged by the sensor, leaving gaps in the coverage; and
• foreground objects obstruct the scanner’s view of features of interest, leading to
gaps in the data coverage.

A typical MTL point cloud of urban features is presented in Figure 1-1. Features in the
scene vary across a range of spatial scales and are seen from the sensor at different angles.
Similar features are often sampled at very different densities as a factor of feature geometry,
distance from the sensor, and speed of the system when that feature was sampled.

Figure 1-1. The data product of a LiDAR scan, aptly called a LiDAR point cloud. These images show the
southeast side of Macintosh-Corry Hall, home to the Department of Geography at Queen’s University. Top:
MTL point cloud collected using the TITAN® system. Bottom: LiDAR returns processed to show data
collected from Macintosh-Corry Hall. Note the gaps “cast” by obstructing trees.
In short, urban scenes are complex, as is the LiDAR data. A number of these challenges arise from the unique point of view of the scanner relative to scene features. Complexity is also caused in part by the level of detail and number of different objects in the scene.

1.6 Geographic Information

MTL units such as the TITAN® system (described in Section 2.11) provide data at the scale of large everyday urban objects. Objects at this scale are commensurate with direct human experience of urban space (Couclelis 1992). The “within-scene” point of view is familiar; so one way to make sense of MTL data of urban scenes is to draw on knowledge derived from direct human experience. The focus of one component of this research project is not to extract or classify any one particular feature from an MTL point cloud but to explore frameworks for improving these processes through better visualization, more meaningful data representation, and a strong set of urban scene heuristics.

Overall, the viewing perspective MTL has of an urban scene is fundamentally different from airborne LiDAR and is responsible for the challenges outlined above. An aerial sensor by definition observes the scene from afar whereas an MTL system collects data that often surrounds it. This unique perspective of the scene limits the transferability of established aerial LiDAR methodologies to the MTL domain. In addition, the problem domains are very different as the desired outputs from these are distinct.
1.7 Levels of Detail

In addition to location of features or objects, the ultimate goal is to have objects with semantics, in other words, to have both geometry and description. There is a movement in GIS towards grounding spatial data with semantics first and geometry second, e.g., via a vocabulary for storing and querying city models (Couclelis 2010, Zeigler et al. 2000, Frank 2007, Stadler & Kolbe 2007). CityGML® is an example of an established standard schema for storing and sharing 3D city models (Kolbe 2008). The schema includes five levels of detail ranging from landscape surface models (LOD0) to models of building interiors (LOD4). The detail of the physical models increases, as does the thematic or semantic granularity. At medium level, features include the interactive or architectural components such as windows and doors. These levels roughly coincide with the data density provided by the three different types of LiDAR acquisition systems currently used in urban contexts (see Figure 1-2), although the acquisition geometry profoundly affects what is actually captured as described above. MTL captures a level of detail comparable to CityGML® level 2. For example, Malambo & Hahn (2010) use MTL data to refine level 2 models and propose its use for creating level 3 details in a CityGML® model. However, the actual use of MTL data for creating features at that level of detail is tool-dependent and often requires use of a number of different software packages throughout the feature extraction workflow.
Figure 1-2. City Levels of Detail and their correlation with LiDAR data collection practices.

**LOD0**
- Regional
- Elevation Model
  --Major topographic features

**LOD1**
- City
- Block Buildings
- Major infrastructure
- LOD of most current city data

**LOD2**
- City Block
- Buildings with
  - Differentiated Rooftops & Photorealistic Façade
- Minor infrastructure

**LOD3**
- Site
- Exterior Architecture
- Infrastructure components

**LOD4**
- Building information + Interior structure
1.8 Research Goals

The goals of this thesis are twofold: first to investigate the sensor perspective for making sense of MTL data and second, to develop methods for feature extraction based on semantics and urban ontology. This research examines a number of methods for representing and manipulating MTL data for the purpose of extracting information. This is done through the development of new methods for experts to inspect and process data. It is posited that a view along the MTL system trajectory – a LiDAR scanner viewshed – is useful for representing, understanding, and analyzing urban MTL data. Further, object-oriented analysis has a role in understanding and classifying data seen from this perspective. To this effect, three aspects of spatial data analysis are explored: (i) data representation using geovisualization techniques; (ii) object-oriented workflows for feature extraction; and (iii) subsequent representation of features in a GIS.

1.9 Thesis Outline

Chapter 2 provides an introduction to LiDAR technology and mapping in general and specifically in the context of MTL. It begins with a description of LiDAR and the technologies that are used to support LiDAR-based mobile mapping, with an emphasis on the urban environment. The TITAN® MTL system that acquired the data used for this research is described in detail. Finally, current challenges for the adoption of MTL are discussed within the context of this research.

In Chapter 3 the geovisualization aspects of MTL data are considered. The limitations of current tools for processing MTL data are described. A new tool that takes the LiDAR scanner point of view on the urban scene into account is introduced along with recommendations for useful visualization techniques for augmenting MTL data representation in virtual environments.
Processing of MTL data is considered in Chapter 4, where a system using the semantics of urban spaces and predictable geometries of urban features is described and the results are processed for use in a GIS. Again, the LiDAR scanner perspective is adopted to improve data representation.

The final chapter (5) presents a discussion of the utility of the methods developed for extracting urban surfaces and structure from MTL data and provides recommendations for further research within this forum of mobile mapping.
2.1 LiDAR and Supporting Technologies

LiDAR uses a laser beam to derive distances to objects/surfaces and thereby determine positions of those objects/surfaces. Contemporary LiDAR systems are capable of taking hundreds of thousands of accurate distance measurements every second. The scanners can be stationary or a component of a mobile mapping system where the sensor and controls are mounted on a vehicle, in which case they need to know their position when each laser pulse is transmitted and received. Mobile systems use a survey-grade differential GPS system (DGPS) to keep track of the trajectory of the system while scanning, and an inertial measurement unit (IMU) to monitor smaller and higher frequency changes in acceleration and attitude. Specialized hardware and software of the data acquisition system are needed to control and monitor data flow from each component (for an overview of state-of-the-art mobile systems see Shan & Toth (2009)).

For mobile system, the scanner, DGPS, and IMU data are used to generate a georeferenced set of points referred to as a LiDAR point cloud (Figure 2-1) that includes Easting (X), Northing (Y), height (Z), and intensity (I) for each point. LiDAR system components are described in greater detail below.
Surveyors and planners were the initial adopters of LiDAR technologies because: (i) the accuracies from static systems can satisfy survey standards; (ii) data acquisition is fast; and (iii) they capture sufficient information to move much of the existing survey work into a virtual environment. In other words, accurate measurements can be made in a software environment using data collected in the field.

![Figure 2-1. Example of a typical urban street corner point cloud collected using the TITAN® MTL system in Kingston, Ontario, Canada.](image)

### 2.2 Laser Ranging

Laser ranging can be performed in one of two ways: (i) measuring the pulse time-of-flight (TOF); or (ii) measuring the phase-shift of a signal (Ackermann 1999). These two systems differ in their range precision and data collection frequencies. TOF sensors transmit a laser pulse and record the time it takes for the pulse to return to the sensor using highly precise timers and the speed of light to calculate the distance travelled. The phase-based approach uses a full-waveform beam as opposed to laser pulses. Changes in the phase of the signal, caused by interaction with surfaces, are combined with the duration of the signal to calculate distance.
(Amann et al. 2001). Phase-based sensors are inherently more accurate than TOF sensors but have shorter range. Typically, TOF sensors are able to log returns from distances up to 1500m whereas phase-based systems are limited to a range of approximately 70m due to phase-ambiguity at distance.

2.3 LiDAR Scanner Components

LiDAR scanners use small circuit-controlled motors, or servos, to precisely change the direction, or azimuth, of the LiDAR beam when scanning using mirrors. The actual path the beam follows is set by the combined motions of the motors and, in the case of a mobile system, the mobile platform itself, which results in a complex pattern of “sampling” of features down-beam. As described below, precise timers are also needed so that data can be merged with positional data from GPS / IMU and other sensors such as CCD arrays (see Mrstik & Kusevic 2009).

2.4 Positioning Technologies

Mobile systems rely on highly accurate positioning systems with centimeter-level precision. In mobile systems it is necessary to be able to accurately locate where the system is for each LiDAR return since any error will be added to the uncertainty of positions returned by the scanner. LiDAR systems use beam orientation, offset distance, and scanner geographic position on the Earth to calculate georeferenced locations for each LiDAR return. For a static system, this amounts to DGPS: which uses the carrier-phase of the GPS signal as well and local base stations of known location to get a very accurate position of the scanners location. For a moving system the issue is more complex since DGPS requires calibration and monitoring to assure good positional data. DGPS positions are only obtained once a second (termed epoch) whereas LiDAR
sensors are sampling thousands of times a second. Since there can be significant movement of the system between GPS epochs, data from an inertial measurement unit (IMU) helps to interpolate positions to fill those positioning gaps (Figure 2-2). In addition, IMU can determine the 3D pose of the sensor assembly, while the GPS cannot.

**Figure 2-2.** A mobile system uses accurate positioning technologies. The internal measurement unit (IMU) fills in the gaps between the GPS epochs and provides pose. Timing is the key to merging data from various sources.

The key to positioning with GPS is calculating the time that it takes for a signal to travel from a set of GPS satellites to the GPS receiver (VanSickle 2008). To do this a GPS receiver compares its internal coded signal to the coded signal from GPS satellites. The difference between the two signals, or phase-shift, is used to calculate the distance from each satellite (VanSickle 2008). The quality of GPS signals changes over the course of a mobile survey because it depends on satellite geometry (i.e., their relative positions in the sky) and interference caused by objects between satellites and the GPS unit, such as buildings and large trees.
A DGPS corrects for errors in the positioning of the vehicle during operation due to poor satellite geometry or obstruction of signals. DGPS uses base stations at known static locations that have clear unobstructed views of the sky and thus the GPS satellites. Variations in the positions collected by the always-static base stations are anomalous and so can be used as corrections to improve the positioning of the mobile system (Figure 2-3).

In addition to the code-phase component of the GPS signal used by common GPS units, GPS receivers used in mobile mapping also make use of the carrier-phase of the GPS signal (VanSickle 2008). The carrier signal that carries the timing signals is transmitted at a higher frequency, meaning it is a finer resolution signal which makes discrimination of time more accurate, leading to more accurate distance calculations, often accurate at centimetre scale. Regardless, while GPS provides absolute positioning every second, the truck moves during that time and therefore finer positioning, as well as pose information, are required.
2.5 Inertial Measurement Unit

The inertial measurement unit (IMU) measures micro-scale accelerations and changes in attitude at approximately 2 KHz. This data is used to interpolate how the system is moving in space (i.e., change in X, Y, Z and pitch, yaw, roll) between carrier-phase GPS measurements. The IMU is connected rigidly and as close as possible to the scanning unit in order to minimize differences in their relative positions during operation as this can be a significant source of error (Glennie 2009). The IMU needs to be very accurate and precise because small errors in attitude (angles) are exaggerated over distance (Figure 2-4).

![Figure 2-4. IMU Error Amplification. The IMU keeps track of small changes in position (X, Y, Z) and attitude (pitch, yaw, roll) of the system. Accuracy is important because small errors in angle measurement are exaggerated over distance such that pose errors have much higher impact than position errors.](image)

2.6 Data Acquisition System

The GPS, laser scanner and IMU are individual components producing multiple data streams. The Data Acquisition System (DAS) is a hardware and software system built to capture and store digital data from these coupled systems as well as to provide monitoring of data quality during acquisition. Data are being received from each component at different rates. Some data may also be processed inside the DAS, e.g. correction of the positioning via DGPS or more advanced merging of data in real-time (Mrstik & Kusevic 2009). As such, DAS need to be designed around data flow as shown in Figure 2-5.
The combination of LiDAR scanner, DGPS, IMU, and DAS is a system of hardware and software components that are used to track the acquisition of LiDAR data in real time and log that data for post-processing into a georeferenced point cloud. In real-time, the error attributed to satellite signal quality and drift in IMU calibration between DGPS updates is measured by the system and monitored by a crew working inside the acquisition vehicle.

![Diagram of mobile system architecture and data flow](Adapted from Glennie 2009)

**Figure 2-5.** Example schematic of mobile system architecture and data flow. (Adapted from Glennie 2009)

### 2.7 LiDAR Products and Application

The point cloud is the set of recorded returns by the LiDAR scanner. In the case of mobile systems, the returns are georeferenced with easting, northing, height and intensity. A single survey can result in billions of individual points – several gigabytes of data. This data is used to produce a variety of data products and there are various tools available to facilitate this.

Value is derived from a LiDAR survey when LiDAR data are used to produce useful geographic information. At the landscape scale, high-resolution digital elevation models (DEMs) are interpolated from LiDAR to support road planning, forest management, and geological mapping, to name a few. In the urban context, LiDAR data are used in virtual survey applications.
for site design and ongoing monitoring or to model heritage sites. For example, timely cut and fill estimates provide excavators with information to support more efficient digging and leveling.

LiDAR data supports the extraction of building rooftops for updating geospatial databases and city visualization (Vosselman & Dijkman 2002, Lafarge et al. 2008). At the scale of MTL data, features of interest in the urban scene are tagged with meaningful information, such as the location of utility poles, their type, and mounted infrastructure. In many cases, derived products consist of two-dimensional (2D) lines or areas, such as the road edge and shoulder, much more compact than the raw 3D data. MTL LiDAR data are being used primarily along transportation corridors to provide higher detail surface models and for infrastructure assessment. For the most part, feature extraction to points, lines, and surfaces is done semi-automatically using specialized software.

The types of products and what features can or cannot be resolved depend on how the LiDAR data was acquired and the perspective of the system relative to features of interest. This varies depending on the type of LiDAR system.

2.8 Types of LiDAR Systems

LiDAR sensors are deployed as standalone static systems or are integrated into a more complex system with positioning technologies on a mobile platform as described above. MTL systems bridge the gap between airborne and static terrestrial systems.

2.8.1 Static Terrestrial LiDAR

Static systems are typically used to capture features of interest in high detail. To capture large features such as buildings often requires taking multiple scans from different locations
because portions of target features are occluded from the scanners point of view by obstructing foreground objects or the geometry of the target feature itself. Separate point clouds can be stitched together using a set of common reference points and georeferenced by mapping at least three known locations in the point cloud. Static scanners have the advantage of precision and affordability.

2.8.2 Airborne LiDAR

Airborne systems remain the most common form of LiDAR survey because they provide coverage of large areas where the point of view gives data ideal for generating high resolution digital elevation models. This perspective coincides well with the established working environment in GIS and remote sensing. Aerial LiDAR systems typically use a single scanner that has an oscillating or rotating mirror to capture surfaces in a swath below the aircraft, and the flight path is chosen based on desired point density, where point density increases with each successive swath overlap.

2.8.3 Mobile Terrestrial LiDAR

MTL systems are similar to aerial LiDAR systems requiring accurate positioning coupled with rapid data acquisition but generally require two or more LiDAR sensors in order to capture the geometry above, below, and to both sides of the platform. MTL systems solve some of the limitations of static systems in acquiring street point-of-view data and provide higher density point data than is collected from the air in terms of efficiently capturing urban areas.

MTL provides the means to survey urban scenes from the ground perspective at flow of traffic speeds. An external frame containing two or more scanners, digital cameras, a DGPS
receiver and an IMU is mounted above a vehicle. A DAS and computer control system is
installed inside the vehicle where a crew monitors the positioning quality and system operation
during acquisition.

MTL data are georeferenced using custom post-processing software as part of the survey
process. The interpolation of the system trajectory takes positions and pose of the system into
account, projects the scanner point data using the positions and pose along the trajectory, and
estimates error in the acquisition. The post-processing is rigorous and takes substantially more
time than is available during the acquisition itself.

MTL provides data at intermediate density and coverage in relation to airborne and static
LiDAR data acquisition systems. Compared to airborne LiDAR, data density for the road near the
vehicle is higher and vertical surfaces occluded in the aerial perspective are captured. Compared
to static LiDAR, a mobile platform captures overall geometry quite well, but provides lower
sampling density.

In short, MTL is a complimentary LiDAR technology that augments airborne LiDAR
data along transportation corridors and can be used as an accurate reference to georeferenced
static scans. For example, if an area is not captured at sufficiently high density, a static scanner
can be used to fill in detail. The georeferenced MTL data makes it much easier to align the static
scan to a geographic coordinate system (Figure 2-6).
Figure 2-6. Data fusion of Leica HDS6000 static scanner data to the TITAN® point cloud. The area shown is at the intersection of Clarence St. and King St. in downtown Kingston, Ontario. Note the high resolution Whig Standard building and the lower resolution MTL data for adjacent buildings providing the context in downtown Kingston, ON, Canada (Image courtesy of Matt Lato, Queen’s University).

2.9 LiDAR Processing Software

As mentioned above, LiDAR data offers volumes of point locations tagged with intensity, and colour information provided the system is integrated with a camera system. Aside from the processing of data inside the DAS and post-processing that produces a high-quality, georeferenced point cloud is the process of actually deriving features and geographic information using other software tools.

Software tools for processing LiDAR point clouds come in different forms where the perspectives on the data, modes of operation, and combination of automatic and semi-automatic methods available vary. Many of these tools are built to support processing of airborne LiDAR data, for example Bentley Microstation® and Merrick Mars®. More recently, limited LiDAR
analysis support has appeared in GIS software, typically focused on processing airborne data. In other cases, 3D virtual environments (e.g. a CAD-like environment such as Innovmetric Polyworks®) are tooled for working with scans of individual parts in industrial applications. Other similar virtual environments such as Leica Cyclone® are aimed at virtual survey but processing tools often assumes data from a static scanner.

Workflows for working with MTL data make use of all three types of LiDAR software. There is a lack of available software specifically tooled towards processing MTL data. Neither airborne nor static scan-oriented software performs well, and tools such as Polyworks® are unnecessarily complex. As a result, there is currently a bottleneck in terrestrial LiDAR use: it is easier to capture data than to get value from it.

2.10 The Mobile Mapping Industry

MTL systems have advanced rapidly over the last decade. Today, MTL systems are commercially available and provide high accuracy solutions at increasingly affordable prices (see e.g. Optech 2007). Currently, LiDAR surveying is a service industry where independent firms are contracted to collect LiDAR data and then to post-process it into usable information. The types of features extracted from MTL LiDAR must fit into existing spatial data infrastructures, almost always based on GIS, which were not built to include analysis of point clouds or true 3D data. Thus there is a gap in the capability to use LiDAR data to produce spatial information on an ongoing basis, and acceptance and innovation by end-users has been limited.

LiDAR acquisition systems are becoming more affordable and full survey-capable systems are appearing on the marketplace (e.g. Optech Lynx Mobile Mapper®, StreetMapper®). It is not inconceivable that mid-sized cities will eventually see the value in repeat LiDAR survey
for infrastructure management and will move towards conducting these surveys themselves. This gradual expansion of use is very similar to the diffusion of GPS technology in the 1990’s. As of yet, and as discussed above, the corresponding shift in supporting analysis tools has not occurred.

### 2.11 The TITAN® MTL System

TITAN® is a MTL solution developed by Terrapoint / Ambercore Inc. in response to the need for a reliable mobile mapping system for transportation corridors. This thesis was funded as part of a joint project between Queen’s University and Terrapoint / Ambercore Inc. aimed at examining improved workflows for use of MTL data for feature extraction in support of urban mapping and gaming applications. TITAN® was used to acquire the data herein and knowledge of the TITAN® system setup was used for the development of custom tools and methods for processing LiDAR data.

TITAN® is capable of multi-orientation scanning while driving at the flow of traffic speeds. It can be mounted on different vehicles but is typically attached to an adjustable boom mounted to the frame of a pickup truck (Figure 2-7). The system uses a sophisticated DGPS linked to multiple base stations, an IMU, and four Riegl® model LiDAR scanners (Glennie 2009). At the time of this writing, each sensor operates at 10 kHz for a combined operating scan of 40 kHz, or 40,000 points per second. Two side-mounted sensors point to the side and slightly forward with respect to the platform. Two additional scanners point upwards and downwards to the rear respectively, scanning in the across-track direction. The scanners use rotating mirrors and have a swath width of 80°.
Figure 2-7. TITAN® and acquisition vehicle. The TITAN® pod is mounted atop an adjustable lift and data cables run into the vehicle cabin where the crew monitors and controls the DAS. Here TITAN® is shown installed on a truck equipped to drive on railways. (Image courtesy of Craig Sheriff, Ambercore Inc.)

A representative view illustrating the operational situation is presented in Figure 2-8. The system captures the road behind the vehicle very well. The side scanners are positioned such that they capture the vertical surfaces of roadside infrastructure and building façade parallel to the system’s trajectory. Due to the speeds at which the scan takes place, surfaces that are roughly parallel to the beam, i.e. at high incidence angles to the travel corridor, have very low sampling densities. Some returns are also recorded from surfaces inside buildings. In some cases, scans are collected in both directions on streets to minimize geometric occlusion and to maximize point densities.
The TITAN® system has particularly good positioning accuracy, on the order of 5 cm absolute and 2.5 cm relative for areas at typical urban street distances (Glennie 2007), i.e., drift in GPS position during the acquisition can cause overall positioning error to be as much as 5 cm whereas the accuracy for smaller sections or areas of the scan will be much lower. Recently the TITAN® system has undergone a number of modifications, including the addition of a camera to colour the point cloud (Mrstik & Kusevic 2009). Colour information is useful for segmentation of the point cloud into objects. These enhanced data were not available for this research.

2.12 Summary

LiDAR provides volumes of distance measurements from surfaces within view of the scanner. There are three types of LiDAR data acquisition methods: static, airborne, and mobile-terrestrial. There are fundamental differences between them given their different perspectives of the scene and the gap between data acquisition and processing methods. This thesis addresses the gap for MTL and for the TITAN® system in particular. Two approaches follow: the first focuses on the visualization of MTL data to end-users and the second focuses on representation of MTL data in GIS and development of automated data processing workflows.
Chapter 3: Geovisualization Techniques for Mobile-Terrestrial LiDAR: A Sensor Model Perspective

3.1 Introduction

MTL is poised to play a major role in the future of urban mapping. Segmentation and extraction of features from LiDAR point clouds has applications in 3D city modelling, planning, and disaster management. The need for high-resolution models of urban corridors is driven by dynamic modern urban development where precise geo-location and imaging of urban features such as utility poles and road signs for planning and city infrastructure maintenance programs are of interest.

Workflows for feature extraction from MTL at these scales are in their infancy. MTL systems acquire data over relatively large areas and thus capture geo-location data for a wide variety of urban features, meaning that a wide variety of extraction approaches may be required. Visualization tools are especially important to LiDAR data workflows because of the need to make sense of large volumes of raw, unstructured data.

Here, we explore the LiDAR sensor perspective on urban structure as a way of visualizing and understanding urban structure via MTL data. It is posited that since MTL data
capture is fundamentally different from other LiDAR systems, it can benefit from techniques that take those differences into account. A tool for visualizing MTL data of urban and suburban corridors from the LiDAR sensor perspective is presented as a preliminary step towards developing more efficient workflows for extracting urban geographic information from MTL data. Conceptually, the tool re-creates the data acquisition process in a virtual environment and augments the display with visual aids that better communicate MTL data for the purposes of extracting geographic information.

Workflows for segmentation and feature extraction are often specific to the method of LiDAR acquisition. As discussed in Chapter 2, there are three main methods of LiDAR acquisition: aerial, static-terrestrial, and mobile-terrestrial. The key differences are point of view on the urban scene and the influence of mobility. The relative distance, orientation, and movement of the sensor with respect to surfaces being scanned affects the distribution of returns and determine which features can be resolved. Methods for extracting features from airborne and static LiDAR data implicitly or explicitly exploit the perspective from which features were scanned. However, this aspect has not been transferred to user-oriented MTL tools.

Aircraft LiDAR surveys capture large geographic areas. While the point density is survey dependent, typical density can range from 0.5 - 5 returns / m². Data is easily incorporated into GIS for analysis because of the ‘top-down’ perspective. GIS is also the dominant computing environment used in the planning community. Feature extraction methods exploit the characteristics of the point cloud that result from airborne scanning (Axelsson 1999). For example, since very few points are recorded on building walls a basic distinction between ground and detached objects (buildings) can be made by testing for steep changes in height from one point to the next (Sithole & Vosselman 2003).
In stark contrast to airborne LiDAR scans, scans from static LiDAR scanners are used to capture architectural detail of individual buildings, to acquire data for virtual surveys, and for modeling industrial infrastructure. Scanners are typically mounted and leveled on a tripod. Data resolution can be so high so as to exceed the beam footprint size (i.e., < 1mm). Analysis is done using specialized software that incorporates knowledge about the sensor location during a scan and assumes a high point density and uniform coverage (Pu & Vosselman 2007). Again, the methods exploit the scanner perspective on the scene. For the most part, feature extraction is done manually with the aid of semi-automatic tools that rely on known scanner position relative to scanned surfaces and, in particular, on approximate surface orientation. Interestingly, while the software tools make use of sensor position, the data is normally presented as a cloud seen from an arbitrary perspective and complex scenes are disorienting: that is, the sensors’ perspective is not used to help the operator.

MTL systems collect data from a moving platform using a system similar to those mounted on aircraft, but possess a within-scene perspective, similar to static LiDAR systems. They are designed to fill the gap between aerial and static LiDAR survey at an intermediate point density and coverage and are typically mounted on an automobile that is driven at traffic speeds. The combination of a moving system and a within-scene perspective gives non-uniform point coverage and complex point patterns. The natural self-occlusion of features, for example, means that objects are often one-sided and present only the “side” facing the moving, street-local sensor.

The movement of the scanner through the scene is captured by the data acquisition trajectory. The georeferenced positions measured by the DGPS unit and the minute variations in attitude measured by the IMU are combined using custom software to calculate a trajectory solution (Glennie 2009). The trajectory gives the locations of the scanning system when each
return was collected as well as the pose of the system, providing the link between the scanner and a global referencing system.

In the approach described below, the trajectory positions are used to place a user in the scene and to trace the path of each individual laser pulse returned to the sensor. The trajectory data provides the necessary information to establish a good perspective for visualizing and extracting geographic information from MTL data, thereby enabling the development of MTL-specific tools and techniques in much the same way that methods are designed to operate on airborne or static-terrestrial LiDAR data.

The central thesis investigated here is that the sensor view is paramount for understanding the spatial structure of MTL data (especially incomplete data with occlusion issues) and that it can be exploited for more efficient feature extraction workflows. This approach must center on the user, not the tool’s algorithms. The use of a sensor-relative representation for MTL feature extraction methods and visualization environments is explored using a custom interface tool programmed using the Processing® environment.

Regardless of the exact method used to collect the data, the normal progression to LiDAR analysis is to segment the point cloud into sets of points corresponding to discrete surfaces or features. Models and related geospatial information are then derived from the segmented point cloud. A set of segmented points can be used to model the surface, volume, orientation, or point location of a feature of interest. In many cases, the result may simply be a database providing features of interest (e.g., pole type and location). The key is having tools that facilitate the extraction of urban features commensurate with the level of detail provided by MTL survey and consistent with how users think and work.
3.2 Feature Extraction

Depending on the level of detail required and the software used, the amount of user interaction with the LiDAR data and tools varies. The actual modes of interaction have a strong impact on tool effectiveness. Broadly, there are three fundamental types of point cloud tools. They include those that support:

1) Manual point cloud digitizing.
2) Heads-up, interactive processing or “assisted digitizing”.
3) Extraction using computer vision techniques or other automated methods.

Digitizing is segmentation of the data by a human whereas segmentation refers to algorithmic techniques. Assisted digitizing is normally limited to click and trace with semi-automatic tools – these are essentially interactive versions of manual digitizing where the operator guides the process by establishing regions of interest and refines parameters interactively. For example, the operator will work by clicking seed points for road growing or power line tracing (e.g. Lam et al. 2010, Jwa et al. 2009). Vendors often rely on a diverse and woefully incomplete set of tools adapted from airborne and static scanning practices to extract geo-information from MTL data. Many projects rely largely on assisted digitizing, especially at the architectural component level (e.g. Kersten et al. 2009, Pritchard 2007). Literature on automated methods for processing MTL data is promising (Zhao & Shibasaki 2003, Fruh & Zakhor 2004, Carlberg et al. 2009) but segmentation is still limited to lower levels of detail than are captured by MTL systems, i.e., methods for classifying individual features such as poles are just beginning to appear (e.g. Lam et al. 2010).

Existing tools do not work very well with MTL data and those that are designed to do so are limited to extracting very specific features. Most actual work involves careful interactive
examination by skilled analysts. Therefore, human interaction tools are an important aspect of producing usable geo-information from MTL data – an aspect that has not been fully explored in the literature.

The tool presented here demonstrates that MTL LiDAR data has potential to model sub-block detail down to interaction components such as doors. It puts the user at the center of the feature extraction workflow by considering the problems of interaction and geo-information extraction workflows from a geovisualization perspective.

3.3 Geovisualization and Human-Computer Interaction

Current tools for extracting features are not designed to work with MTL data and as such can be cumbersome and ineffective. LiDAR service providers and end-users can benefit from improved tools that streamline MTL feature extraction workflows, including improved visualization frameworks. Here, interactive tools that exploit concepts from geovisualization are examined. The focus is on interactivity, i.e., allowing users to develop a comprehensive and intuitive understanding of the spatial structure of their data. In other words, as is common in the geovisualization world, the users’ cognitive strengths are emphasized in the process of turning raw LiDAR data into useful features for modeling and decision-making (e.g. Gahagen 1999, McEachren et al. 2004, Demsar 2009). The focus includes an improvement on workflows for component-level segmentation and feature extraction where perspective views of the data, components of the display, and specific MTL visualization techniques are considered.

This section first provides background to the geovisualization paradigm. Then, MTL data complexity is briefly reviewed. Finally, the idea that LiDAR maps open space in addition to
points on surfaces is introduced along with a discussion of its importance to MTL geo-
information extraction workflows.

3.4 Geovisualization and LiDAR

Geovisualization relies on a cognitive mapping of the properties of data in the mind of
the observer (Gahegan 1999). The presentation of data informs the user about the spatial
structure of the data and provides insight into different ways of working with data to extract
information and to produce knowledge. Specifically, the orientation of the user and the
effectiveness of the within-scene display as a problem-solving tool are considered in this work.
The general geovisualization and visual analytics paradigm is described by Slocum et al. (2009)
and the cognitive basis of geovisualization is explored by McEachren (1995). Note that the
GeoVISTA project (Takatsuka & Gahegan 2002) has data processing workflows and that the
GEON project has explored alternative geovisualization and workflows for airborne LiDAR
(Crosby et al. 2006), but neither consider MTL data or urban areas.

Functionally, a geovisualization approach considers tasks, users, and interaction
(McEachren et al. 2004). Specific tasks performed by specialists are geared towards knowledge
construction and require a higher degree of interaction. More general tasks, such as presenting
and sharing of information require less interaction with raw data and less specialized users. Tools
that display point clouds in virtual 3D environments, some of which are immersive (e.g., Kreylos
et al. 2008), are the norm in LiDAR processing, are usable by experienced operators, but are far
from ideal for visual interaction with raw MTL data. These environments typically focus on
improving the display of large, landscape scale, airborne LiDAR datasets (e.g., LViz – Crosby et
or are designed to work with data collected from static terrestrial scanners (e.g., Leica Cyclone®).

Figure 3-1 includes examples of MTL data in a virtual 3D environment. Variable density data collected from a moving platform makes it difficult to visually interpret or to perform spatial operations on the data. Problems include:

1) Manipulating a 3D view involves six (6) degrees of freedom. Establishing suitable views of the regions of interest takes considerable time, even for experienced users.

2) The point cloud itself does not communicate what regions have or have not been scanned – the distinction is important for interpretation and quality assessment by analysts.

3) Variations in point densities fail the basic data input requirements in available semi-automatic tools, e.g., when growing a planar region the tool requires a minimum distance threshold between points. Set the threshold too low and the algorithm omits points, set it too high and points from adjacent surfaces are erroneously included. Velocity flux during data acquisition is enough to render methods that rely on such parameters unusable.

The research presented here outlines the design of a tool that the user will use in segmentation and feature extraction workflows for MTL data. It is designed in keeping with the geovisualization philosophy where the point-of-view aspect, existing data, and the actual configuration of the sensor are explicitly examined. The aim is to present the data and to design the user interaction in such a manner so as to alleviate the complexity of working with MTL data and to clearly communicate what regions of space were scanned in addition to the location of
point returns. This is done in part by redefining point cloud navigation as a coordinated and goal-directed process following the acquisition trajectory.

Figure 3-1. Existing visualization perspectives for a TITAN® LiDAR point cloud of upper Princess Street in downtown Kingston, Ontario, Canada. *Top and Centre:* 3D virtual environment (Merrick MARS®) perspectives looking down the sidewalk and a bird’s eye view from above the scene. The curve shows the acquisition trajectory. *Bottom:* Top-down view of MTL data in ArcMap® GIS.
3.4.1 Data Complexity

Complex point distributions result from the within-scene perspective and movement of the sensor through the scene. For example, data density diminishes with distance, i.e., the sidewall in Figure 3-2 has a much lower sampling density than the front wall, and the non-linear diagonal scan lines are caused by the relative surface-beam geometry as the sensor is moved past the structure. If the sensor was static, there would still be density drop-off with distance, but the lines on the sidewall would be nearly vertical and the effects of surface-beam geometry would be less pronounced.

Figure 3-2. Point density and point patterns are a function of distance and movement of the sensor. The point density diminishes with distance and point patterns vary on a wall that is nearly parallel to the scanning plane. While the arrow indicates the sensor perspective of the house, the sensors’ position is not intuitively obvious from the points alone.

Visualization methods that show how these point patterns are created can help analysts understand what level of detail they can reasonably expect from available data and help in planning future acquisition routes. Gaining an intuitive understanding of the processes that led to the point distributions potentially offers insight into new automatic processing methods and improved workflows.
3.5 Open Space

An often-undervalued source of uncertainty in LiDAR data is unmapped space resulting from occluding surfaces (Yapo et al. 2006). Reciprocally, LiDAR proves that there is a path of open space between the point location and the sensor. As a result, LiDAR technology potentially provides volume maps in addition to point locations, in particular maps of open space. Conventional environments make no distinction between what space was or was not traversed by a LiDAR beam. While the actual logistics of mapping open space from LiDAR are scale-dependent and can be complex at the microscale, the work presented here results from ideas about mapping and using information of open space inferred from the MTL sensor perspective.

In urban GIS and planning, visibility analysis aims to capture and to quantify the viewable urban space from locations in the urban scene. Termed viewshed, isovist or visualscape, the visible space is structured by form and spatial configuration of elements and is thought to affect human cognition of and behaviour in architectural (Benedikt 1979, Hillier 1996) and urban (Batty 2001, Llobera 2003, Turner et al. 2001) spaces. Such approaches are being extended into 3D with the availability of accurate 3D urban models (Yang et al. 2007).

In projects that hope to capture urban form and features, it is useful to make a distinction between open space and occluded space for two reasons. Firstly, unmapped space is a key factor in post-mission quality assessment to ensure that there was adequate coverage of the features of interest. An understanding of the effects of occlusion can also aid in mission planning. Secondly, MTL data is truly 3D, meaning that obtaining geographic information for volumes of space is possible by leveraging human knowledge of the general structure of urban space. For example, an analyst can make informed decisions about the extent of a building and some of its internal...
structure from images like the one presented in Figure 3-3. The analysts’ knowledge about houses would be able to tell them that these points are associated with a house and that there were returns recorded through windows and most likely from interior walls and ceilings. Those features are not quite so obvious when only the points are shown. While the LiDAR scanner may have only recorded one or two passes of the beam over these interior surfaces an analyst can come to quickly recognize this pattern as window openings and interior rooms. The analyst can also infer that these spaces are rectangular and create those features accordingly – filling in some of the unmapped space (i.e., showing open space increases the amount of information presented).

![LiDAR beam passed through the shaded region. This tells us something about where walls can and cannot be.](image)

*Figure 3-3. Visualizing Open Space. Arrows indicate the sensor perspective on the data. Clearly, the position and size of different sections of scanned space helps differentiate features by providing additional context.*

Here, we are examining ideas for mapping and using information of open space inferred from the MTL sensor perspective. It turns out that these ideas have also been explored in the robotics community: Tracking open space from active sensors comes from research on
maintaining maps for autonomous vehicle navigation, known as Simultaneous Localization And Mapping (SLAM), where distinguishing between what has been mapped (and what is not) is important for navigation (Matthies & Elfes 1988, Pagac et al. 1996). These ideas are extended for use with LiDAR sensors in autonomous navigation by Montemerlo et al. (2006) and for constructing maps across multiple scans by Yapo et al. (2007).

3.6 A Visualization Tool: MTL Viz

Sensor models have proven successful in processing active ranging data in real-time (e.g., Petrovskaya & Thrun 2009, Harati et al. 2007). The sensor perspective is not normally exploited in LiDAR visualizations or in the construction of a data model for analysis. Here, it is hypothesized that the sensor perspective is crucial to developing an understanding of the spatial structure of an MTL dataset. The methods outlined here model the sensor robotics and the mobile systems location in the scene. The result is new frames of reference that are more natural for viewing and working with MTL data. A simple MTL data interaction framework based on the trajectory is presented. First, the implementation is briefly described. Then, a comprehensive overview of the frames of reference used for presenting data to the user, the modes of interaction, and the visualization techniques are provided. Using the tool as documented below, a skilled analyst will be able to do the following processes that are currently difficult with existing tools:

a) Have an intuitive perspective of the scene.

b) Analyze spatial structure of urban areas using useful / unique diagnostic perspectives.

c) Deal with the confusion that large MTL point clouds present when using traditional tools.
3.6.1 Implementation

The MTL visualization tool is implemented using the Processing® visualization development environment (Fry 2008, Reas & Fry 2007). Processing® is a Java™ development environment that includes libraries and functions to enable the rapid development of visualization tools and applications. For example, abstract functions are tasked with setting up the screen and for drawing useful primitives in 2D and 3D. Processing® compiles the Java™ code and launches the application using a ‘Play’ button that is included as part of the development interface. This setup is designed to facilitate rapid prototyping for visualization projects. The integrated compile-run-debug process speeds up the test cycle. Full source code is given in Appendix A.

The advantages presented by a rapid prototyping environment are that there is the flexibility to try different approaches to visualization, such as changing the colour, opacity, or number of elements displayed to the user. However, this is a non-standard approach and requires the time to learn how to program in a particular language and environment. Alternatively, GeoVista Studio (Gahegan et al. 2002) provides a visual design interface for developing interactive geovisualization applications that does not require coding but does require significant training time. Processing® was used because of the need to manipulate and create data, to produce non-standard views, and provide more control of user interaction methods specific to the design of the visualization tool.

The resulting application can be deployed as a stand-alone application or an embeddable web applet. This means that the program can be downloaded and run on an end-users computer or loaded on-demand inside of a Java™-enabled web-browser. The objective was to efficiently explore the latitude of potential visualization tools, not to develop production software.
3.6.2 Frames of Reference

In the developed tool, data is represented relative to sensor location along the trajectory. Buildings and other urban features often align well with the roadways and this perspective is also familiar to analysts. Two complimentary 2D representations are used (see Figure 3-4):

a) Along-Track (looking along the trajectory)

b) Across-Track (sensor look direction relative to the trajectory)

![Along-Track](image1.png) ![Across-Track](image2.png)

**Figure 3-4.** Along-track and across-track perspectives of the same LiDAR points as seen from the LiDAR scanner, relative to the data acquisition trajectory.

Reducing the number of dimensions needed for display and analysis is made possible by considering a single scanner perspective. Moving to a 2D representation removes a major barrier to analysis of LiDAR data in current GIS and remote sensing software and removes the added complexity of displaying and interacting with 3D data on a 2D display.

Representative examples of the two perspectives are shown in Figure 3-5. The along-track perspective presents the data as a cross-section of the corridor. The across-track perspective
presents the data as though one is riding on the acquisition vehicle and always looking in the direction of the sensor.

Local dimensions are derived from LiDAR data: the distance of the sensor from the X and Y coordinates (i.e., Easting and Northing), the distance along the trajectory, or along-track distance (ATD), and the height and angle relative to the sensor (Figure 3-5). Overall, three dimensions are used to display the data: a 2D display with navigation along the trajectory.

Furthermore, note that three different representations for the vertical axis are considered: height from the geoid; the point height relative to the sensor height; and the angle formed between the point and sensor positions and a level plane at the sensor height (Figure 3-5).

**Figure 3-5.** Various metrics used for the sensor frame of reference. The image shows the TITAN(R) system mounted on a pickup truck and the relationship between the beam path and alternative height metrics.
Option 1 is useful because it is the actual geographic height while options 2 and 3 are unique to the MTL perspective. Option 2 is in keeping with a sensor-centric visualization and option 3 is suitable because it eliminates overlapping points by spreading the data out ‘as seen’ by the LiDAR scanner. All three options can be made available to users of a visualization-based software tool but for the purposes of this chapter, height from sensor is used throughout.

At first, this approach may appear to throw away much of the advantage of the 3D data; however, since cities have predictable geometries and the orientation is chosen to match those, exactly the right 2D orientation on the geometry is displayed to meaningfully understand and extract those 3D features. Note that these methods would break down should one scan an area where building geometry relative to the road was highly inconsistent or where vegetation or other obstructions was sufficient that the point cloud captured little of the building structures. Either of these situations should be detectable by the operator.

### 3.6.3 Modes of Interaction: Location, Scope, and Scale

The system trajectory data is used as the basis for placing the analyst in the scene. Arbitrary views are complex and there is an advantage to taking a slice of the data, especially if the orientation simplifies the view. It turns out that the ideal orientations are those as discussed above. What this means is that the tool orients the data for the analyst. Moving along the trajectory replicates the motion of the sensor as it was moved through the scene.

The analyst navigates by moving the sensor through the scene along the trajectory. Creating geographic information at the scale of MTL data necessitates localized views. In current tools, navigating the virtual environment and hiding extraneous data from view are advanced and time consuming prerequisite operations. The trajectory gives the local views and the order in
which the returns were logged. Relevant surrounding data are found in the immediate block or slice of data.

In computer graphics terms, we have shifted from navigating with six degrees of freedom to one degree of freedom – i.e., along the path. Now, the relevant interaction is navigating to the local region of interest and establishing the appropriate scope and scale. The analyst jumps rapidly to a position along the trajectory using a scrollbar that represents the entire trajectory or can step through the dataset, refining the exact position using the arrow keys or a mouse wheel.

Changing the step size affects the apparent navigation speed through the scene. As the region of interest becomes narrower, the user will need to step through the points using smaller steps. Therefore, the step-size button is programmed so that the step size increases by a factor of 10 at each successive scale; allowing the user to change the step size by a single point between 1 and 10, by 10 points between 10 and 100, by 100 points between 100 and 1000, and so forth. This approach aims to reduce unnecessary interaction by providing the user with a functional, predefined scaling scheme.

In addition to refining the step size for navigation, scope or viewable width displayed data can be refined by the analyst. This is accomplished by changing the slice width of data along trajectory (in metres), relative to the current position as illustrated in Figure 3-6. Reducing the slice size removes clutter, allowing the analyst to focus on very thin slices of data. Increasing the width of the slice adds context that helps to recognize a feature in the scene.
Finally, once the region of interest along the trajectory has been established and refined, the user may need to change the scale or ‘zoom level’ of the data slice. The LiDAR scanner on the TITAN® system records returns at distances out to approximately 50m while building facades are on the order of 5-15m in height. Scaling the scene allows the analyst to zoom out for additional context or to zoom in to increase the level of detail. And of course, the entire scene can be translated around the display screen.

Overall, locking the analyst to the trajectory gives excellent perspectives on the MTL data and establishes a conceptually simple basis for navigation and view refinement. Next, visualization techniques that help the analyst understand the point distributions and nature of MTL data are discussed.

### 3.6.4 Visualization Techniques

The frames of reference and modes of interaction described above use the ordered trajectory data to guide the analyst. The techniques described below augment the point cloud display by modelling the path of the laser beam through the scene and showing the user what
space was scanned by the system (or not). To do this effectively requires modelling aspects of LiDAR scanner operation.

3.6.4.1 Modelling Scanner Operation

Special consideration was given to the actual operation of the robotic system that oscillates the transmittance of the laser pulse. Some systems, like Optech’s Lynx Mobile Mapper™ collect over 360° while others, like the Riegl® Q120 scanners used on TITAN®, oscillate to collect series of scan lines. In this case, when the beam changes from a very negative angle (i.e., pointing at the ground) to a large high angle (i.e., pointing up towards a tree branch) a break point is created between successive scan lines.

Here, scan lines are especially important because they provide an additional level of organization that is exploited when showing regions of scanned versus open space to the analyst.

To best represent data along an individual scan line requires modelling more than just the breakpoint between scan lines. The angular resolution of the scanner determines the precision with which it measures the angle of the beam relative to the entire unit - thus determining the maximum resolution. In this case, the angular resolution is 0.4° across an 80° swath. Modelling these aspects of system operation is crucial to fully communicating what is actually scanned. For example, knowledge of the angular resolution and swath width is used to fill in missing laser beam paths at the top of each scan line where there is often nothing but sky, and so no returns.

Eventually, the idea is to be able to distinguish between shots where there was no return because of empty space, and those where there was no return because the beam was reflected or refracted by glass or high incidence angles with a very smooth surface, such as the hood of an automobile. This characterization can help to build augmented visualizations that give much more
information to analysts that in turn results in more efficient workflows and better information products.

3.6.4.2 Augmentation

A simple augmentation in the tool for providing meaningful context is to draw the line between the sensor and current point location as a thin red line. Here, a small black circle, indicating the location of the currently focused point in the data, terminates the line. Although selection of points is not available in the current version of the tool, explicitly navigating through the data by focusing on individual points offers an advantage over selection using a cursor / mouse. It can be difficult to select the correct point at larger scales where points are very close together on the screen. On the other hand, context is lost when selecting an individual point using a cursor at small scales. This is why scales of navigation through the data are important. The analyst can move point by point at a reasonable rate in order to pick an individual point while still retaining a zoom level that provides sufficient context.

A line between the sensor and point is helpful but is rather static in that it is showing a snapshot of the acquisition process. There is movement and directionality to MTL scans. Two complimentary techniques that communicate scan directionality are considered. The first technique is to apply different colouring or shading schemes to points that were scanned after the current position in the data – or are “about to be scanned” – and those that are ‘already scanned’, relative to the current position. As illustrated in Figure 3-7, blue colouring identifies points that have already been recorded, up to the current position, and green colouring indicates those points that the system will acquire further along the trajectory. This is in keeping with the idea that the tool recreates the scan process for the user as a part of the visualization environment.
Furthermore, points close to the current position are completely opaque while those further away are increasingly transparent. This gives a fade-in/out effect while moving through the scene, mixing context with clarity.

Figure 3-7. Point placement relative to the current focus. Different colouring or shading is applied to point behind and in front of the current position to communicate directionality of the scan.

The second technique draws more than just the beam path of the currently selected point, and, in fact, begins to reveal more about scanned versus occluded space in the scene. This technique also relies on a proper modelling of the scanner robotics discussed above. Points are processed in overlapping pairs along an individual scan line. The triangle formed between the two points and the sensor is drawn to the screen. To show directionality, the triangle associated with points further behind the current point in the dataset increase in transparency (Figure 3-8). This technique not only maintains a consistent gradient showing scan direction towards but also communicates more about what space has been scanned by the scanner, as per the recorded returns.
Figure 3-8. Trailing, semi-transparent triangles show directionality of acquisition and communicate scanned space in the scene.

This idea is further extended to show the open space along each scan line collected by the system. Recall that a scan line is a sweep of the sensor across the scene in one direction and that knowledge of the LiDAR scanners mechanics used to point the beam as well as its operating resolution and swath provide the necessary information to determine what portion of space was scanned.

Rather than drawing individual triangles along the scan line, a polygon is formed by the points along the entire scan line using the sensor as the start and end points to close it off. There are often no returns recorded from the beginning (top) of the scan line because the beam misses the top of buildings and there is no return to record. Therefore, the scan line is reconstructed in the open space where no returns were recorded using knowledge that the system collects over an 80° swath with steps approximately every 0.4°. An operating range of 50m is used as the maximum range within which open space has been proven – resulting in a smooth arc. The algorithm for drawing open space polygons was designed to take the pitch of the scanner into account by considering the last point in the scan line and then back-calculating the angle of the initial shot. Figure 3-9 illustrates the open space across three consecutive scan lines. Note that
three open-space polygons are displayed with linearly decreasing opacity levels. It was found that displaying more than one open-space polygon was more aesthetically pleasing and clearly communicated local open space.

Figure 3-9. Open space polygons constructed from the point set and knowledge of scanner-specific robotics.

The design goal was to communicate what was scanned, or not, in the scene. Point clouds alone fail to communicate that volumes of open space were scanned and proven empty by the LiDAR system. The complexity of urban scenes leads to complex occlusions effects. Drawing open space makes it intuitively obvious what was and was not recorded by the scanner. Methods for displaying scanned open space were tested and built on the underlying framework for drawing successive laser beam paths. Such methods need additional algorithms that incorporate a set of heuristic rules that model the robotic aspects of the LiDAR sensor itself. Communicating the acquisition helps users understand the point distributions that they see on the screen. The primary concept is to display the path of the laser beam between the surface and the sensor. Visualization methods to communicate the path of the LiDAR beam through the scene as well as the order of data acquisition were tested. Showing a single beam path in the scene sometimes lacks context.
Showing a number of preceding paths, as scanned, communicates the nature of the acquisition process more clearly. The Riegl\textsuperscript{®} scanners used in the TITAN\textsuperscript{®} system scan in one look-direction as a series of lines. Hence, segmenting the points into scan lines was an important step towards producing meaningful visualization techniques that communicate the data acquisition process and effectively reveal street-level urban structure.

3.7 Example Scenario

In this section a hypothetical example implementation of the tool is presented. “The city” is interested in monitoring property frontage and sidewalk conditions in different neighbourhoods. The scenario depicts a case where an analyst has just received a LiDAR point cloud from an acquisition completed the previous day. She is tasked with evaluating the occlusion of sidewalks and lot frontage caused by roadside objects – mostly vehicles, in MTL scans. Using MTL Viz, the analyst produces a report for a subsample of streets following the process outlined in Figure 3-11. The scenario is given as an example of what can be done with the tool in its current form. It demonstrates how navigation through the data works in principle and a scenario where displaying open space is useful.
3.8 Limitations

MTL systems are operated in complex urban environments where they are typically restricted to roadways. In addition, various mounting geometries exist for different types of scanners. The MTL data representations presented in this chapter were designed to work for a single MTL system. Here, some general limitations of the methods with regards to MTL systems are outlined in conjunction with some system-specific issues.

Figure 3-10. Example workflow with MTL Viz.
3.8.1 Reversed Scan Order

The methods presented may not be suitable for analyzing sections of the data where the vehicle travelled in a tight arc, causing the point samples on the inside of the arc to become compressed and eventually overlap while the points on the outside of the arc become more dispersed (Figure 3-12).

![Top-Down View](image)

**Figure 3-11.** The effects on data distribution when the MTL system is travelling in an arc. The order of returns traces backwards across the building wall to the inside of the arc and will be retraced as the system continues moving down the street.

Return number 1 and 2 in Figure 3-12 come from somewhere down the street. Points 3, 4 and 5 on the inside arc are actually being collected in the opposite direction from the scan and as the system proceeds, those areas will be retraced by the LiDAR beam. In this way, ordering based on location along the trajectory can give confusing results when using an across-track display.
3.8.2 Mount Geometry and Pointing Systems

The visualization and representations of MTL data explored above were designed for data collected from a single Riegä» Q120i LiDAR scanner mounted with a specific geometry relative to the direction of travel. This scanner uses a rotating prism that passes the beam in one direction over the scene (top to bottom for the side-looking scanner considered in this chapter). Other LiDAR scanners have rotating or oscillating beam directions. Using data collected from a LiDAR scanner with different mechanics and mounting geometries would require a modified visualization approach tailored to those specific systems. For example, the Optech Lynx Mobile Mapper™ system uses two rotating scanners so a sequential scan possesses a helical structure. Despite different scanners and mounting geometries, the basic principles outlined here should still apply.

Similarly, for the 2007 TITAN® system, the visualization can faithfully represent any two of the paired scanners, left/right or up/down, in a single image. However, due to scanner mounting geometries as shown in Figure 3-13, that shows data from the rear-downward and sideways-left scanner, there is an offset in real-world positioning that would need to be considered. This incongruence can be eliminated with some preprocessing of the data but this has not yet been implemented.

Figure 3-12. TITAN(R)® sensor offset between side and rear-facing scanners. Top View.
3.9 Conclusions

Workflows for feature extraction from MTL are a relatively new area of research. MTL data exhibits different characteristics than aerial and static LiDAR making it difficult to incorporate into existing workflows and processing environments. In the research presented here, the potential of the sensor-relative perspective was explicitly examined from a geovisualization paradigm. Visualization and user interaction components of working with MTL data were explored using custom-built visual tools in the Processing® environment. A minimalist set of user interaction methods were implemented for moving through the point cloud while maintaining a consistent working view for the user. A sensor-model was used to construct a representation to communicate the nature of the data collection methodology and the complex point patterns that are produced as a result. Visualization methods were designed from the sensor model towards explicitly mapping what space was scanned and what was not; a crucial and useful distinction that captures knowledge of open versus occluded space in urban scans.

Other similar approaches to feature extraction for MTL data focus on the initial segmentation of features and ultimately remove the user from the operational workflow. By contrast, this work focuses on the specific needs of the user in the workflow process. As such, the focus was on representations that minimized the effects of data volume and complexity.

The general sensor-relative paradigm for semi-automated feature extraction from MTL data was explored in this research. A simple transformation of the data produces meaningful representations that have potential to improve understanding of complex LiDAR point clouds and of feature extraction workflows. Future research should aim to test for improvement by comparison with other workflows that use traditional 3D environments. A study comparing the
time and quality of feature extraction at different levels of detail (i.e., ranging from buildings to building components) would be of interest. For example, studies can be designed to investigate where bottlenecks in the digitizing process occur and whether or not these can be resolved by automated or semi-automated methods.

Additional features should provide for the capacity to overlay or integrate existing spatial data. A pragmatic application of such data integration would be, for example, to correct the boundaries and height information for building polygons and provide higher levels of detail to existing urban models.

Finally, given current and future MTL development, there is a requirement to incorporate additional sensors into the visualization process. However, to do so effectively requires a more sophisticated treatment of data alignment from different sensors. This represents a focus for further research.
Chapter 4: An Object-Oriented Approach to Extracting Geographic Information from Urban Mobile-Terrestrial LiDAR Data

4.1 Introduction

As discussed in Chapter 1, high-resolution 3D mapping data of urban areas is important for establishing the nature and spatial distribution of land use, including mapping of transportation corridors and related infrastructure. MTL is an emerging technology that collects high-resolution data about the geometric distribution of objects along transportation corridors. It has direct application in urban model development for city infrastructure mapping and assessment, urban simulation and gaming, large-scale heritage documentation projects, and urban analysis at finer scales than those currently feasible with data from airborne and space-based remote sensing systems. In this chapter, an innovative representation of MTL data for analysis is presented and object-oriented methods for extracting features are developed.

MTL acquisition systems represent a response to the gap between the wide coverage of airborne LiDAR systems, and static terrestrial systems that have limited coverage but provide very high-resolution data; i.e., they capture an intermediate resolution with good coverage in a timely manner. The technology is most appropriate for use in urban areas where vertical surfaces are of significant interest, e.g. building façade and signage. The wealth of additional information
that is obtained from the MTL viewpoint has yet to be fully exploited. Efficient, robust, and flexible methods for extracting information from MTL point clouds are needed to examine potential mapping and analysis products that support the study of urban areas and systems.

As an example of recent urban analysis, the Defense Advanced Research Projects Agency (DARPA) outlined the Urban Reasoning and Geospatial Exploitation Technology (URGENT) program. The goals were to develop 3D urban object recognition systems for planning and situational analysis during war-fighting missions (DARPA 2007). The “Objects of Interest” outlined in the program consist of 150 objects (e.g. fire hydrant, free-standing sign, bridge, gate, and door) spanning seven categories: urban entities, sites, transportation, marine objects, building components, terrain, and vegetation. The use of contextual knowledge-based approaches and methods insensitive to occlusion in point clouds are listed as particularly desirable outcomes. The focus is on situational awareness, where location and accuracy of the classification are more important than photo-realistic geometric models. Such approaches require high-resolution data and strong workflows to organize and model scene-knowledge for feature extraction.

Work in machine vision for autonomous robots/vehicles are able to generally classify hazards and make sense of urban corridors in real-time. In the work that follows, aspects of methods in this and related fields are borrowed and an approach using an object-oriented tool for building feature extraction procedures is evaluated.

There are similarities between the goals of the URGENT program described above and what purveyors of LiDAR data need to accomplish to deliver quality products. Efficient feature extraction methods and the production of usable geographic information from LiDAR point clouds are central themes. A LiDAR scan results in a set of points typically referred to
collectively as a point cloud that will represent incomplete and uneven sampling of urban features. A key aspect of recognizing features is segmenting the point cloud into meaningful subsets that, hopefully, correspond to actual objects or perhaps components of larger compound objects.

What exactly constitutes a meaningful segmentation is application dependent. Part-whole relationships often need to be defined along with a strong set of semantics. For example, is a telephone pole with transformer one or two objects? In short, feature recognition requires a strong ontology of features, or objects in the space. Ontology is a particular conceptualization of reality (Guarino 1998). It is a system of related concepts, attributes, topological and taxonomic relationships of the kinds and structures of objects in reality (Smith 2003). Ontological descriptions of the environment have been used in robotics systems for a number of decades, motivated by models of human cognition of the environment that suggest a hierarchical organization that is largely semantic in nature (Kuipers 1994). In the geospatial context, ontology is seen as a way to capture greater meaning, of providing a map between different geospatial practices and conceptualizations at a high semantic level (Tanasescu 2007). Here, urban domain ontology is used to capture the overarching concept of spatial relations and semantics that form the generic building blocks for the ontological language. Ontology is an abstraction paradigm, mapping the human-cognitive universe to a representation in a computer system, and domain ontology exists between the general concept of geographic space and usage-based analysis (Fonseca et al. 2002).

In this research, the ontological language consists of the procedures, functions, and rules available in Definiens’ eCognition® software. Recognition routines rely on good segmentation and ways to map ontologies onto objects through statistical and contextual analysis. The
following section explains the object-oriented approach to image analysis from an urban geography perspective.

The wider adoption and use of MTL data is limited by the processing and analysis capacities in existing remote sensing and GIS to work with MTL data. Recently, modules for working with airborne lidar data have been introduced in popular GIS’ (ESRI ArcGIS®, GRASS, Definiens’ eCognition®). However, the 2.5D nature of such systems is not useful for MTL data where hundreds of points can be present for a single 2D location. 3D-capable point cloud processing tools are better suited to MTL processing tasks but are cumbersome to use because most are intended for working with terrestrial scanner data or with scans of individual mechanical parts. Here, a method for working with 3D data in a 2.5D object-oriented software tool and an automated feature extraction process designed for knowledge-based urban-scene models are presented.

4.2 Background

Exploiting the characteristics of the data acquisition process and scanner configuration relative to the scanned scene has proven to be a fruitful strategy in processing MTL data. Fruh and Zakhor (2004) segmented the scene using histogram filtering and subsequently built ground and textured façade models. Zhao & Shibasaki (2003) exploited scan-line data along with RGB information from a line scanner to reconstruct textured façades and ground in a similar manner. The information was then used to update a geospatial database. Carlberg et al. (2008) reconstructed urban scenes through a mesh-adaptive structure where the MTL point cloud is triangulated as a series of scan lines. They used volume-based (i.e., voxel) methods to adapt surface meshing thresholds to regions with different point densities. Their method offers
significant improvements over existing surface reconstruction methods, demonstrating the applicability of scan line based approaches and the need for MTL specific methods. Hernandez and Marcotegui (2009) also exploited range images and applied the Top-Hat hole filling algorithm to identify artifacts on road surfaces collected by an MTL system. Moosmann et al. (2009) project static terrestrial data onto a cylinder aligned with the rotational axis of the LiDAR scanner. They construct a mesh structure in a similar manner to Carlberg et al. (2008) but apply a four-neighbourhood graph, which is similar to raster structure, and demonstrate that this representation essentially reduces the problem of 3D segmentation to 2D and that good segmentation results can be achieved in urban environments using their ‘Local Convexity Criterion’. Thus, there are a variety of different strategies for segmenting terrestrial data but the general theme remains that aspects of the data collection process and sensor itself help to simplify the process.

Strategies that do not exploit the characteristics of the scan acquisition process rely on additional data and processes to establish meaningful context for feature extraction. Kim and Medioni (2010) apply a tensor voting approach to 3D object segmentation in airborne and mobile terrestrial point clouds. The combination of airborne and MTL data provides the necessary context to first identify the major structural components of a scene (ground and buildings) and then to move towards segmenting and classifying smaller urban objects. Their method also takes the variable density of MTL data into account by using a volume-based spatial index at multiple scales. The multi-scale nature of MTL point clouds occurs both as a result of point density and the variable sizes of features found in the scene.

Vosselman et al. (2004) provide a comprehensive review of segmentation methods for structure recognition in airborne and static terrestrial LiDAR. Segmentation is done based on
threshold conditions or can be multi-stage where the properties emerging from initial segmentation results are used to merge similar regions. For example, topographic features such as the width of height contours or the direction of linear features can be incorporated into merging rules. Generally, strategies assume some type of smooth or planar surface. Conditions for merging adjacent points or pixels rely on neighbourhood relationships and thresholds for distance and difference in surface orientation (Jiang & Bunke 1994, Vosselman 2003). That is, the planar fit for a point is computed and merged with neighbouring pixels whose planar equations are similar. A fundamental overview of surface-based segmentation methods is provided by Hoover et al. (1996).

Working with MTL data in existing geospatial software is fraught with difficulty. Despite the relative successes of methods for modeling the geometric structure of urban surfaces and the general segmentation and classification of façade and ground surfaces, paradigms and frameworks that facilitate MTL analysis, exploration, and methodological experimentation are rare. Currently, custom frameworks and algorithms are created to address very specific feature extraction problems and since these are research-only solutions, MTL vendor and end user needs are not addressed. These frameworks are unfamiliar to most people who work with geospatial data, particularly those who do not work in production settings. This raises the question of whether production tools – tools that are used in production settings – can be adapted to perform feature recognition for MTL data.

4.2.1 Object-Oriented Image Analysis and Urban Scene Models

Object-oriented image analysis is emerging as a promising paradigm for the analysis and classification of high-resolution remotely-sensed imagery. As opposed to pixel-based image
analysis, object-oriented image analysis considers segmented regions of an image as the basic units of analysis. This approach has been successfully used to segment and extract features from airborne LiDAR data (Blaschke & Hay 2010, Brennan & Webster 2006, Walker & Blaschke 2008). The research reported here was carried out to investigate the methods and processes involved in analyzing MTL data using a commercially available object-oriented image analysis tool, i.e., Definiens’ eCognition® v 8.0.

Object-oriented design concepts provide a more natural and flexible manner for describing image data (Mohan & Kashyap 1988). Object-oriented image processing places image-objects (image-segments) into class hierarchies, for example lawn and sports field are both subclasses of grass, and into spatial hierarchies that capture composition across spatial scales where, for example, a wall is composed of bricks (Figure 4-1). This approach of describing a domain of things and the relations between them is implemented by object-oriented programming languages where the building blocks of the ontological language constitute procedures, functions, and rules (Tanasescu 2007).

4-1. Image-Object and Urban-Class Hierarchies.
Definiens’ eCognition® exposes class and image-object hierarchies to the user but hides the database and programming aspects of the system. Figure 4-2 shows the user interface elements available to the user. The metrics, or attributes, of any individual image-object are displayed in a window when that object is selected. The viewports can be configured to display a raster layer, image-object metrics, classification, and augmented with image-object boundaries. The user assembles a custom feature extraction procedure at a high level by adding image segmenting and classification algorithms, or modules, to the ‘Process Tree’. Rules are defined within each module to determine the domain of ‘image-objects’ that operations should be performed on. For example, an operation could be designed to merge image-objects that have high intensity returns and are within 2m of a building. A second module can then classify the resulting image-objects as a highly-reflective street sign.

Figure 4-2. Definiens’ eCognition® user interface. Viewports display various views showing classification or image-object metrics that are in turn used to develop the rules and procedures that make up the process tree.
The object-oriented approach allows users to build flexible semantic models and to explicitly specify relationships between those objects (Mohan & Kashyap 1988). It affords imagery analysis, or in this case, LiDAR data in raster form, using geographic concepts such as object location (absolute and relative to other image-objects), their size and shape, and their neighbourhood. Rules encapsulate the conceptual constraints about what objects are found where, and in relation to what other object. In GIS terms, this is a combination of raster and vector-based paradigms. In more general terms, it strengthens the connection, or mental mapping, between understandings of reality and how it is represented; enabling design of urban scene ontology.

The object-oriented image analysis workflow is typically described as a linear process that begins with segmenting the image into meaningful ‘image-objects’. An ‘image-object’ is the name given to a segmented region of pixels in the overall image; rules define conditional parameters that determine which image-objects are considered for manipulation or classification. In light of actual cases, it has been recognized that the process is iterative and involves multiple stages of merging, splitting, and subsequent reconstitution of image-objects (Baatz et al. 2008). So, different segmentation strategies work better for targeting certain features in the scene. To get the best representation of image-objects might require more than a single segmentation and consideration of different combinations of information layers. Therefore, overall the analysis process or workflow, depicted by the flow diagram shown in Figure 4-3, is actually assembled from multiple procedural components.

The actual process will vary significantly from project to project. The key is that rules are used to narrow the domain of image-objects to be manipulated or classified. There is significant latitude in ways to narrow the domain by selecting image-objects with specific characteristics for analysis, manipulation or classification. The domain of possible image-objects is narrowed by
considering class membership, feature values relative to neighbouring object (topology), internal composition of sub-objects or pixels (mereology), as well as shape, size and location.

Figure 4-3. Object-oriented workflow. An initial segmentation into image-objects is followed by many iterative steps of image-objects modification and classification before exporting usable geographic information.

Rules can be complex, combinations intended to capture general knowledge of urban scenes. For example, we know that the approximate distance between the LiDAR scanner and the ground is 3m and that the road extends no more than 8m laterally from the scanner. Since cars are usually found above the road, only image-objects less than 3m below and within 8m range are treated as potential cars.
The set of rules model the expected structure of an urban scene, hence a *scene model*, which is synonymous with the general meaning of the term as used in recent literature (e.g. Kim & Medioni 2010, Lam *et al.* 2010, Cornelis *et al.* 2008). The scene is reconstructed starting with the features deemed easiest to identify, providing a general context, which is subsequently used to make better decisions for recognition and classification of other features in the image. Feature extraction methods that take this approach are broadly called *knowledge-based* and have strong roots in shape grammar for design (Kristic 1999, Stiny 1972). Pioneering work using static LiDAR scans has demonstrated its effectiveness (Haala & Becker 2009, Pu & Vosselman 2009). The reader is directed to Mayer (2008) for a comprehensive overview of object-oriented feature recognition and to Haala & Kada (2010) for a thorough review on the current status of automatic building reconstruction methods.

A main concern with development of automatic systems such as the one presented here is reliance upon a large number of tuning factors where specific approaches work well on certain sites but are not widely applicable. Urban scenes have some advantage in that individual city districts can have highly regular geometries. When represented from a consistent perspective, i.e. the sensor travelling along a transportation route, the complexity of establishing a suitable representation for analysis is reduced.

**4.3 Presented Work**

As noted above, the work here examines object-oriented analysis workflows on MTL data using production tools from a remote sensing perspective using Definiens’ eCognition® v8.0. This investigation includes two stages: data preparation and analysis.
Methodological issues commonly cited in the literature are concerned with gaps in the data due to occlusion and point density variance of MTL data. The approach taken herein differs from previous work on MTL feature extraction in that it takes a sensor-centric perspective on the dataset while maintaining access to the georeferenced location data (Easting, Northing, Height, or [X Y Z]). The trajectory information is used in conjunction with the raw point data in order to transform the data into a sensor and trajectory-relative coordinate space in preparation for object-oriented analysis. Use of the scanner characteristics (i.e. angular view of the scanner rather than georeferenced height) in representing the point data has the advantages of removing occluded regions from the analysis domain and reducing the effects of surface-beam geometry on point density because the data is presented as the sensor saw it (Figure 4-4). The approach is also practical; the transformation puts the point data into a 2D image space suitable for analysis using widely available GIS and remote sensing image analysis software.

An object-oriented approach to semi-automatic feature extraction from the transformed LiDAR image is then applied. The goal is to investigate methods towards use of semi-automated or automated object-oriented rule-sets for extracting features from urban MTL data via commercial tools. The approach aims to capture urban heuristics using contextual, knowledge-based methods. Both the spatial relations between objects (topology) as well as information about their makeup (mereology and statistics) are used to provide rich descriptions that drive the feature extraction process.
Figure 4-4. Top: Data is represented in height from scanner (left) and height from geoid (right) for two different sections along Princess Street in Kingston, ON. Note the viewpoint data gaps in the top-left image caused by foreground objects. Bottom: Data along the vertical axis is displayed by angle relative to the scanner. Occlusion caused by foreground objects is hidden when considering the scanner-perspective, remaining data gaps (vehicle windows and sky) are regions where a beam was pointed but there was no recorded return.

The suitability of the object-oriented image analysis approach to MTL data analysis is evaluated within the confines of Definiens’ eCognition® tool. This work exposes some of the challenges in using this approach and, more generally, for the construction and use of urban scene models. Here, preliminary thinking about rules for organizing measured urban structure at the sub-block scale (features in the urban corridor) is presented. Exceptions to the rules and difficulties in the methodology are instructive when thinking about how to build and improve future GIS that will need to process MTL data.
The presented approach takes the sensor perspective on the scene into account. We make
the assumption that local road surfaces are not steeply sloped and that building geometry is
captured well from the scanner perspective. The general strategy is to break down the major
components in the scene in the initial segmentation, and then assign meaningful class names to
image-objects based on their relative locations in the scene and general knowledge of urban
structure.

4.4 Methods

4.4.1 Study Area and Data

This research constitutes a component of ongoing efforts to understand urban space for
use in disaster management, infrastructure renewal, and urban gaming applications. The test
target is the downtown core of Kingston, Ontario, a city with a population of approximately
120,000 (Figure 4-5). Kingston includes architecture ranging from circa 1850 to contemporary,
with the urban core dominated by 3-4 story brick and limestone-based buildings built between
1900 and 1950 with some newer concrete, brick & glass structures. Wired infrastructure is for the
most part pole-mounted. Street furniture includes hydrants, postal boxes, benches, signage, and
utility boxes. Ground vegetation is rare near the street but on larger lots can be dense and obstruct
structures from being scanned. The city has an ongoing GIS programme and much of the city has
been mapped to better than 10m accuracy in 2D; parts are available at better than 1m but few
features are identified at this scale and there are significant institutional barriers preventing the
generation of a holistic spatial and semantic urban model.
Data was collected in summer 2007 in downtown Kingston by Terrapoint Inc. (a division of Ambercore Inc.) using the TITAN® mobile mapping system. The author supervised this data acquisition. The system is capable of multi-orientation scanning while driving at flow of traffic speeds.

The system uses DGPS linked to multiple base stations, an IMU and four Riegl® model LiDAR scanners mounted on an adjustable lift and attached to the flatbed of a half-ton pickup (Figure 4-6). Each sensor has a swath width of 80° and operates at 10 kHz for a total combined operating scan of 40 kHz. A scanner points left, another right, and two to the back. The two side-mounted sensors point to the side and slightly forward with respect to the platform. The two additional scanners point upwards and downwards to the rear and scan in the across-track
direction. A representative view illustrating the operational situation is shown in Figure 4-6. The side scanners are positioned such that they capture the vertical surfaces of roadside infrastructure and building facades perpendicular to the system’s trajectory.

The system captures the road behind the vehicle very well. Due to the speeds at which the scan takes place, surfaces that are roughly parallel to the scanning beam have very low sampling densities. The beam penetrates glass and some returns from building interiors are recorded. In some cases we chose to have the scan travel in both directions on streets to minimize geometric occlusion and to maximize point density.

**4.4.2 Data**

As discussed in Chapter 1 and 2, LiDAR point clouds traditionally consist of a list of X, Y, and Z coordinates and backscatter intensity. The data provided by Terrapoint Inc. also included the location of the TITAN® system itself for each returned point. This additional
information is required to transform the data into a sensor-based coordinate space using simple
trigonometric relations.

A subset of the data from one side-looking scanner, spanning two city blocks, and
consisting of 0.5 million points was used for this study. Accuracy assessment carried out by
Terrapoint Inc. estimates that the accuracy of the scan data is on the order of 5cm absolute and
2.5 cm relative for the study area. The section of the urban street scene used has buildings ranging
from single to multiple stories, of which only the first 3 are captured by the side looking sensor.
There are trees, traffic lights, vehicles, streetlights, pedestrians, and a variety of street level
objects in the scanned zone.

4.5 Presented Approach

The general strategy and workflow is summarized in Figure 4-7. First, the raw MTL data
is transformed into a sensor-based coordinate system where the trajectory forms the baseline for
further operation and analysis. Multiple raster images are interpolated from the transformed point
cloud and are imported into Definiens’ eCognition® software for feature extraction.

Feature extraction takes the form of iterative segmentation and classification of image
objects to refine and improve the representation. Sections of the process that find candidate
image-objects and refine their representation are referred to as recognizers. A recognizer
encapsulates the rules for reducing the spatial and semantic domain under consideration, applies
segmentation routines and labels (classified) urban features with semantics.

The product is a classified image and image-object database representing features in the
scene. This information is further processed within a GIS to create geographic information. The
classification can also be mapped back onto the raw points using GIS overlay operations.
4.6 Preprocessing Data in a GIS

Definiens’ eCognition® v8.0 operates on data in raster format. Hence, to perform object-based analyses, the point dataset is transformed into a raster format. This section describes the transformation of raw data into a trajectory and scanner-relative coordinate space and the subsequent interpolation of raster layers.
4.6.1 Transforming Raw Data into Scan Coordinates

The raw MTL data is first transformed into scanner-relative coordinates: the X-axis is cumulative distance along the trajectory and the Y-axis is the scan-angle above or below a level plane placed at the sensor's location. The point-sensor coordinate pairs are used to calculate the range, height and angle of the sampled point location relative to the sensor position as illustrated in Figure 4-8.

Redundant points that have the same X, Y, Z coordinates are removed (typically about 3%) and a new set of transformed coordinates is computed from the raw data. This transformation places points relative to the positions of the scanner along its trajectory through the scene.

![Figure 4-8](image)

**Figure 4-8.** Information derived from the scanner-point coordinate pairs: Height From Sensor (HFS), 2D distance (2D Range) from sensor and the angle from level at the sensor height (theta).

The axes in transformed coordinate space are non-isometric, i.e., they differ in measurement units (radians vs. metres). There are also differences in precision (1/1000th of a radian vs. 1/100th of a metre) and resolution (0.007 rad vs. 0.02 m). This difference affects interpolation because it leads to more analytical weight being assigned to the axis with higher resolution, or more dense point spacing. Therefore, to minimize this bias, the axes are scaled to
match precision and resolution. Figure 4-8 illustrates the rescaling of axes for the sensor-relative system.

The Riegl® LiDAR scanner used for the TITAN® system has a swath width of 1.4rad and an angular resolution of 0.007rad, giving 200 discrete sampling angles. While the GPS trajectory solution has centimeter-level precision, the actual spacing is affected by the fluctuating speed of the acquisition platform. Analysis of the data reveals that there are 12497 unique positions along a 297.69m cumulative trajectory, which is on average 2cm between positions, but the spacing from one position to the next is highly variable as shown in the histogram values listed by Table 4-1.

<table>
<thead>
<tr>
<th>Distance (cm)</th>
<th>Count (positions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5069</td>
</tr>
<tr>
<td>2</td>
<td>4598</td>
</tr>
<tr>
<td>3</td>
<td>83</td>
</tr>
<tr>
<td>4</td>
<td>152</td>
</tr>
<tr>
<td>5</td>
<td>623</td>
</tr>
<tr>
<td>6</td>
<td>309</td>
</tr>
<tr>
<td>7</td>
<td>176</td>
</tr>
<tr>
<td>8</td>
<td>549</td>
</tr>
<tr>
<td>9</td>
<td>341</td>
</tr>
<tr>
<td>10</td>
<td>220</td>
</tr>
<tr>
<td>11</td>
<td>293</td>
</tr>
<tr>
<td>12</td>
<td>79</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>12497</strong></td>
</tr>
</tbody>
</table>

A 7cm resolution was chosen to account for variable spacing between scan lines and above the stated 5cm absolute accuracy of the data. Measurement precision is matched by scaling the angular resolution up by a factor of 10 (0.007 $\rightarrow$ 0.07) and then both axes are divided by 0.07
so that points have an even spacing (Figure 4-9). Scaling in this manner maintains the resolution of the angular measurement while reducing the resolution of the along-track axis to 7cm.

![Figure 4-9](image)

**Figure 4-9.** The axes are scaled for interpolation. **Top Left:** Scan lines shift to the right and have large gaps from one scan line to the next because they were acquired from a moving vehicle. **Top Right:** Matching precision and resolution of the non-isometric axes gives a more uniform point distribution suitable for interpolation. **Bottom:** Points along approximately 20m of trajectory in scan coordinates. Note the gaps in point coverage.

### 4.6.2 Constructing the Analysis Mask

A return is not logged when there is no object within the active range of the scanner or when the interaction between the beam and the surface causes the beam to be reflected in such a manner, that the power of the signal is too low to be recorded. Thus, an analysis mask is used to nullify regions of ‘No Return’ in exported raster layers.
A Triangulated Irregular Network (TIN) structure is constructed from the transformed point set (Figure 4-10). The TIN triangles are then converted to polygons and the area of each polygon is calculated. A maximum triangle area is set as the threshold for ‘NoData’ regions. Small gap triangles represent regions of contiguous data whereas larger polygons represent real gaps. Manually querying polygons that are obvious gaps in the coverage helps to determine what constitutes a large area threshold. A new dataset is created from polygons larger than this threshold size. The shared boundaries of the remaining polygons are then dissolved and all remaining polygons are converted to a raster with a cell size equal to the interpolated images (Figure 4-10).

![TIN data structure (left) and extracted analysis mask (right). Size of TIN triangles is used as a proxy for large gaps in point coverage.](image)

**Figure 4-10.** TIN data structure (left) and extracted analysis mask (right). Size of TIN triangles is used as a proxy for large gaps in point coverage.

### 4.6.3 Interpolating the Raster Dataset

Six (6) raster images are interpolated: X, Y, and Z coordinates, range from sensor (2D Range), height from sensor (HFS), and intensity in the sensor-coordinate space using the natural neighbour method (Sibson 1981) in ArcGIS® 9.3 (Figure 4-11). The natural neighbour method is an exact interpolator that works well on datasets of variable density and in mixed coordinate
spaces (Watson 2001). It produces a surface that is continuous inside the convex hull of the dataset. Interpolations are based on local weighted averages of points where the neighbourhood is determined by the Voronoi diagram.

![Intensity Image](image1)

![Range Image](image2)

**Figure 4-11.** Interpolated raster images. Intensity *(top left)* and various types of spatial information such as range *(top right)* are interpolated. Left: UTM Easting, Northing, and Height contours are shown.

The natural neighbour algorithm interpolates over the complex convex polygon over the transformed LiDAR data. Ideally, interpolation is done between point samples excluding areas where returns were not logged. In order to account for these regions, the analysis mask was applied when exporting layers.
4.7 Feature Extraction

Each stage in feature extraction consists of a number of sub-processes that manipulate the image-object segmentation and further constrain the image-object domain for analysis. First, the ground and large exterior wall feature components of the scene are segmented using custom region-growing processes. Once the major components of the scene have been segmented, ‘recognizers’ designed to identify and manipulate the remaining image-objects are applied. This is accomplished in two phases: first, to recognize street level objects and second, to recognize features relative to building façades. Finally, the image-object hierarchy is exploited to output an image-object table that is suitable for post-processing in a GIS. The first step in this series of stages is to establish the scene model frame of reference inside the object-oriented environment.

4.7.1 Object-Oriented Environment and Scene Model

Definiens’ eCognition® is object-oriented image processing software (Baatz et al. 2001). Therefore, the raster dataset, consisting of the 6 layers described in the previous section, must be segmented into a set of image-objects, where image-objects, as opposed to individual pixels, are the basic unit of analysis. Initially, the raster dataset (hereafter referred to as the ‘image’) is segmented into the smallest image-objects possible – consisting of a single pixel. This segmentation represents the initial starting point for the overall process.

The process outlined here makes use of a large collection of custom image-object metrics that capture relationships between image-objects by class, location, and relational measures. All distance-based metrics such as length, width, height and their derivatives are based on the X, Y, and Z input data. Custom metrics are also used to help establish the scene model context (Figure 4-12).
Figure 4-12. The Sensor-Perspective Scene Model. Schematic indicates key features where the height of the ground and distance to facade relative to the scanner are important.

Trajectory / scanner relative metrics form the basis of the scene model, capturing the distance from the scanner and position along the trajectory. A scanner travel height of 3m above the ground is used as a proxy for local ground height and the ATD and range from sensor are adjusted for the 10° look angle (Figure 4-13). Height from ground (HFG) and the adjusted metrics are exported as additional raster data layers for inclusion in the working project. A complete list of custom metrics used in the feature recognition procedures is given in Appendix B and a list of class specific rules is given in Appendix C.

Figure 4-13. Adjusting for the forward look angle. Top-down view. Note: 10° line angle is exaggerated for illustrative purposes
4.7.2 Large Smooth Surfaces

Ground and exterior wall (i.e., façade) are the largest contiguous features in the urban scene. Conceptually, they are also easy to extract because they have characteristic horizontal or vertical geometries. Moreover, they provide context for more detailed feature recognition by providing the basic framework of the urban scene model. These major smooth-surfaces are grown from single-cell image-objects into larger surfaces following a set of rule-driven procedures that capture their characteristic geometries. Ideally, the fully grown surface would uniquely represent individual wall or ground elements in the real world.

Surface growing consists of finding seed surfaces in the image and then expanding (i.e. growing) the surface outward to include neighbouring objects that meet a set of criteria, or rules. The assumption is that large flat surfaces sampled by the TITAN® scanner are, for the most part, in three different planar orientations: (i) the front façade of buildings that are nearly parallel to the trajectory; (ii) building sidewalls that the scanner sampled at a high incidence angle; and (iii) relatively flat surfaces such as roads and sidewalks (Figure 4-14).

![Figure 4-14. Expected surface orientations from TITAN® scan on an urban scene. This image shows the results of the first feature recognition stage that separates the scene into image-objects with relatively smooth surfaces.](image-url)
Surface orientation is expected to vary smoothly for flat and even for slightly concave or convex ground and wall features in an urban scene. However, calculation of surface metrics, such as slope and aspect, do not give usable results because the data is in a non-isometric coordinate system and the software assumes it to be isometric. As a result, workarounds for segmentation and calculation of metrics were devised.

The expected geometries (i.e. Flat, Vertical Front, and Vertical Side as shown in Figure 4-14) are captured using logical rule sets: flat surfaces should vary smoothly in height where a jump in height signifies an edge, front-facing vertical surfaces should vary smoothly in range from sensor – it is assumed that the MTL system did not suddenly shift sideways during the acquisition, and side-facing vertical surfaces should vary smoothly in the along-track distance because they are roughly parallel to the direction of travel.

Relatively large regions that are classified as image-objects belonging to ‘smooth-surfaces’ are subsequently classified with more semantic granularity as surface, façade, and sidewall. Other ‘smooth-surfaces’ in the urban scene (e.g. access ramps, sloped rooftops, awnings) do not have simple, expected orientations. For example, a different rule would be needed to capture all the differently sloped rooftops – or an algorithm would need to be employed to determine these slopes automatically. Therefore, these features are recognized using a set of rules based solely on a smooth rate of change in ATD, classified as ‘Other Smooth’. This is a consequence of using 2D analysis from a particular perspective as a surrogate for 3D analysis.

Segmenting a scene into surfaces with different orientations is intuitively a simple and practical approach. The actual procedure to segmentation is slightly more complicated and is illustrated in Figure 4-15. Note that the procedure is sequential, that remaining unclassified image-objects get carried forward with each subsequent step, refining the scene model into more
meaningful and consequently, more useful semantic classes. The entire procedure for all feature recognition stages and export of data is given in Appendix D.

Figure 4-15. Large surface-growing process. Flat, Front, and Side surfaces are considered independently. Some image-objects are classified into ground, façade, and sidewall.

Rules R₁–R₃ in Figure 4-15 are each designed to select candidate image-objects for a key surface orientation such as flat or vertical. Candidacy is determined by low mean absolute difference values in height, range, or ATD between an image-object and its neighbouring image-objects, depending on the feature class. At the edges of selected regions of smooth surface candidates are image-objects whose values contributed directly to the candidacy condition but remain unselected. For completeness, those image-objects are subsequently selected as well.

Basic smooth-surface image-objects (i.e., Flat, Front Vertical, and Side Vertical) are merged together into larger image-objects. From these image-objects, those that meet a second set of conditions are classified more specifically as surface, façade, and sidewall following R₅–R₇ in
Figure 4-15. Large flat horizontal surface features at ground level become ground and large front and side vertical image-objects that are more than 8m away from the sensor become façade and sidewall. Size conditions are designed to match with expected sizes of features in the real world: the regularity of the real world is what allows rules to be devised. For example, doors are typically between 2 and 2.5m in height, so as part of the rules for being classified as a façade or sidewall, image-objects need to have a height greater than 2.5m.

Remaining smooth surfaces (i.e., constant rate of ATD change) are merged. This step in surface segmentation not only helps to grow sloped surfaces but also to identify other smooth surfaces that have geometric regularity. A constant rate of change is determined by taking the second derivative of mean difference in ATD to unclassified neighbours (Figure 4-16). Sharp changes in ATD are indicative of a sharp change in geometry. A suitable threshold value was determined by sampling regions of known smooth surfaces in the image such as roofs and access ramps.

Figure 4-16. Second derivative of mean difference to neighbours. Greater rate of change (brighter) signifies sudden positive change in surface orientation, darker signifies a negative change in surface while mid-gray indicates a smoother surface.
Not all image-objects meet the conditions to be classified as ‘Ground’, ‘Façade’, or ‘Sidewall’. Discontinuities occur at the edges between surfaces with markedly different orientations and where surfaces are partially obscured by features in the foreground (e.g., signage, trees). An annealing procedure grows each surface in iterative steps to correct, as much as is possible, for these discontinuities. Surfaces are re-segmented into single-cell image-objects and the neighbourhood is searched for potential candidates that meet merge criteria. Seed image-objects, i.e. those classified as Ground, Façade and Sidewall, are grown to all adjacent candidate objects using image-object fusion. Essentially, this procedure iteratively refines the boundaries of the smooth-surface image-objects using rules similar to the Local Convexity Criterion (Carlberg et al. 2009). The procedure continues until no more image-objects meet the merge criteria. So, the scene is deconvolved into major orientations: flat, front, and side; and more specifically as ground, façade, and sidewall using a rule-driven set of procedures.

4.7.3 Recognizing Small Urban Features

There are a vast number of smaller ‘urban features’ at the sub-block scale. The form and composition of these features varies from city to city, district to district, and even from street to street. Feature extraction at this level of precision may need human operator intervention to capture specific features of interest. Here, specific sets of procedures, termed recognizers, are described for signs, automobiles, steps and curbs. These features are assumed to be pervasive features in urban scenes and to have geometries that can be captured using general and reusable rules relying on form and context. The goal is to demonstrate the use of scanner and urban scene-relative rules to limit the domain of analysis and to classify common large-scale urban features.
4.7.3.1 Sign Recognizer

Signs along urban transportation corridors are usually positioned to be viewable from road lanes but also high enough above the ground to avoid interfering with pedestrian traffic. In the study area, roadside signs displaying parking regulations and street names are placed between 1.8m and 3m above ground. Also, signs are usually highly reflective.

As shown in Figure 4-17, rules can be built which use scanner-relative metrics to capture the expected positions of roadside infrastructure, thereby narrowing the search for candidate image-objects in the scene. The intensity and dimensions of signs is calibrated by reference to actual data.

**Figure 4-17.** Sign recognizer. A traffic sign typically occupies a region of space at a minimum distance from scanner, is expected to be found within a certain height range above the ground, and is associated with high intensity returns.

Rules guide the sign recognition procedure; the full workflow is given in Appendix E. First, neighbouring bright image-objects (i.e., high intensity) are merged together. And second, rules test for suitable position and dimensions (length, width, height) of the resulting image-objects, selecting candidates which are then classified as signs.
4.7.3.2 Step and Curb Recognizer

Some features in the scene can be similar in geometry but differ in their type or use. Steps and curbs have similar height yet are functionally different. As shown in Figure 4-18, context plays a major role in classifying features that have similar geometry. In this case, curbs are mostly surrounded by ‘ground’ image-objects; whereas steps may share some or no portion of their boundaries with the ground, yet are still close to ground level.

Classification rules such those embodied in the Step and Curb recognizer, are conceptually simple and improve the level of semantic description, opening the possibility for new rules, say to help recognize sidewalk, which in turn limits which image-objects can potentially be benches. In this manner a hierarchical classification system can be constructed that eventually recognizes all features of interest in a point cloud.

**Figure 4-18.** Step and Curb recognizers. While similar in geometry, relative proportion to ground differentiates steps from curbs.

4.7.3.3 Automobile Recognizer

There are many automobiles in an urban scene and it is common practice in the LiDAR service industry to remove them from the point cloud, as they are not usually of interest. However, removal implies recognition and there are also applications where vehicle recognition
might be of value, for example to automate parking audits or to estimate traffic density. Vehicles are geometrically complex as they are composed of many different components and some components will not provide quality returns. At this stage in the analytical process, no single image-object corresponds to ‘an automobile’. The side, hood, trunk, roof, tailgate, and wheels are all likely to be represented by individual image-objects. Many of these will have already been classified as one of the three main smooth-surface orientation classes described previously (flat, front vertical, side vertical).

The recognizer described below is also complex. In addition to simple rules about size and position, it is initiated by two different conditions, and has two stages where the result of a merge operation in the first stage establishes the context for the second (Figure 4-19). The first set of rules works in cases where the side of a vehicle was scanned, capturing the size and position of door panels and wheel wells. The second set of rules works in cases where a vehicle was either incompletely scanned or was scanned from the front or rear. In the latter case, highly reflective license plates and reflectors are the initial markers; these are distinct and indicative since they are typically found much closer to the ground than other highly reflective objects (e.g., street signs).

![Figure 4-19. Automobile recognizer. Minimum and maximum height from ground and widths of car side panels define the search space. High intensity markers provide additional cues for poorly sampled vehicles or those scanned from one end instead of the side.](image-url)
The first stage of the recognizer identifies a set of image-object candidates and merges these with any immediately neighbouring flat, front, or side classified image-objects (Figure 4-20). The second stage uses image-object fusion to iteratively merge additional candidate image-objects that fall within the 3D bounding volume given by the Range, ATD, and height (Z) values of the seed image-object. The size of automobiles is used to classify them into 3 sub-classes: car, minivan / SUV, and utility van / bus.

![Figure 4-20. Example of automobile feature merging. Seed (purple) is merged with adjacent flat and side facing image-objects.](image)

### 4.7.4 Bay Recognizer

Methods for extracting large scale scene features ground and façade and smaller scale scene features such as vehicle and poles have been proposed. This section describes methods for recognizing features associated with buildings with an emphasis on recognizing interior volumes, herein termed bays. A bay is thus a depression in the façade structure. Bays can be shallow, such as a recessed entranceway to a retail store, or deep indicating the interior retail space for example. Like initial surfaces and automobiles, bays are grown from seeds that meet specific criteria in a rule set. The procedure is also complex, requiring preparatory stages to set the appropriate context.
and to identify suitable candidates. The recognition procedure grows ‘bay’ image-objects relative to building ‘façade’. Ultimately, such procedures are used to recognize various architectural elements and to build complete descriptions of buildings.

Initially, range-based image segmentation is applied to the remaining unclassified, flat, and vertical objects in the scene and the result is stored as an image-object level above the current segmented image. Then, chessboard segmentation creates square wall objects (approx. 2m ATD in width) for local comparison so that calculations are made relative to local (but somewhat arbitrary) wall segments rather than to entire wall surfaces (Figure 4-21).

![Figure 4-21. Chessboard segmentation of façade image-objects provides localized measures of relative distance.](image)

Transitional zones between features in the scene are identified in the scene and subsequently classified as being part of a bay or a projection relative to the neighbouring façade image-objects (Figure 4-22). Transition zones occur at feature edges, for example where a car boundary in the foreground meets with a façade in the background or around a window edge where the façade feature meets the interior wall that was sampled through the window. A range-based segmentation tuned to break transitional zones into small image-objects and their small size
is used to identify and classify them. Transitional zones that exist inside façade walls have a greater range from sensor than neighbouring façade features and are tasked as seed objects for growing bay features.

![Figure 4-22](image-url) Zones of transition (purple) occur where there is a jump from a foreground object to a surface in the background. Pixels that belong to these zones are identified by a large range difference relative to neighbouring image-objects.

Transitional zones and other image-objects that have further mean range values relative to neighbouring façade features are input as initial seeds to the image-object fusion algorithm. The algorithm iteratively merges unclassified, flat, front vertical, and side vertical image-objects to neighbouring bay image-objects if the result of the merge does not reduce the minimum range value of the seed-image object. In other words, everything inside the building relative the façade is considered to be in the bay. Post-growth rules filter known errors by removing bays that are extremely large or border more than one façade or side-wall image-object.
4.8 Creating GIS Data

Ultimately, recognized features provide value as distinct geographic entities of interest. They represent useful geographic information for use in CAD or GIS for decision-support or as 3D models for visualization and gaming applications.

To transfer results from eCognition® a custom set of image-object statistics is exported to a table that is then used to construct a representation of the scene in a GIS. Features such as signs and vehicles are abstracted to point features placed at their mean XYZ coordinates with dimensions included in the table as attributes. Other features require post-processing of data output from Definiens’ eCognition® software. Details of some feature types are discussed below.

4.8.1 Walls

Walls are simplified to linear features by considering their minimum and maximum X and Y coordinates to reconstruct their vectors. Recall that walls may sometimes be represented by multiple image-objects as foreground objects often obstruct the scanners view. This also means that there will be sections missing. However, the minimum and maximum height values are included in the output table, so even though a section may be missing, the essence of the wall is often retained because upper sections of the wall span the gaps. Once in a vector environment such as CAD, rules similar to those used herein could be developed to repair gaps based on architectural rule sets.

4.8.2 Ground

The ground surface represents the anchoring height for signs, automobiles and walls and is readily represented in a GIS using the TIN data structure. In order to obtain the point locations
to build a TIN, all of the constituent pixel values from ground image-objects are exported and the set is used to construct a TIN in ArcGIS®.

4.8.3 Bays

Bays represent 3D volumes of distinct open space inside of buildings. Each can be associated with a particular façade. However, exporting all the points that are classified as bay, as was done for the ground, does not provide sufficient information to distinguish between individual bays. A novel method for maintaining unique identifiers across the image-object hierarchy was devised to solve this problem. It takes advantage of the fact that bay image-objects at the upper-level in the image-object hierarchy have unique positions. The UTM location of each constituent pixel is exported and is also tagged with the X coordinate of their respective upper-level image-object that serves a unique identifier for sorting the points into sets. A 2D convex hull is then computed for each set. 3D volume information is maintained by tagging each convex hull with the maximum and minimum height values found in their sets.

Ultimately, additional feature recognition procedures could be developed to capture particular features of interest. However, these will be limited by gaps in LiDAR data coverage and level of detail. For example, few returns will be recorded from small features such as chairs and will be indistinguishable. The recognition task is thus limited to features of a certain scale.

4.9 Experimental Results

The above process results in a classified image and a table of selected image-object feature values. The results of processing some of the classified image-object features into GIS datasets are illustrated in 4-23 and Figure 4-24 shows a side-by-side comparison between the classified image and photographs of the scene.
The resulting GIS data comprise of 14 signs, 23 automobiles (15 Cars, 2 Bus’ / Utility Vans, 6 Minivans / SUVs), 51 façade segments, 11 sidewalls, and 340 bays. From site visits, there are 15 signs, 12 distinct buildings, 177 distinguishable bays (156 windows, 21 entranceways). Façade segments represent different parts of a single building façade that are separated by occluding foreground features (e.g. trees, poles) and different façade segments belong to the same building when they have different location, e.g., a recessed entranceway. Separation of image-objects due to gaps in the coverage and occlusion sometimes results in numerous image-objects for the same scene features, e.g., large window and door frames split bays into different segments. The classification results for façade, bays, vehicles, and signs are visually compared to site photographs. Despite a few obvious errors there is good agreement (Figure 4-24).

**Figure 4-23.** Extracted features from object-oriented analysis represented in a GIS environment. Showing the linear wall features, minimum convex hulls from bay-classified point sets, and the mean location of sign and automobile features overlaid on existing GIS data.
Figure 4-24. Comparison of photographs to the scan-image segmentation. The mailbox and disposal bin in the first image set has changed. The awning over the top window in the third set was removed between the time of the scan and when the picture was taken. Vehicles are different as well.
Visual assessment was aided by visualizing the extracted features in ArcScene®, a 3D environment (Figure 4-25). Signs and vehicles are anchored to the ground surface, which is represented using a TIN model and vehicles are roughly oriented to align with the roadways. Façade elements that appear to overlap in the top-down view presented in Figure 4-23 (above) are separated by height and the façade features that they represent become obvious. For example, the tall apartment building has balconies on the 4th and 5th floors. Bay volumes make it intuitively obvious where entranceways and windows provide some sense of the size of interior rooms on the first and second floors.

Figure 4-25. Perspective views in ArcScene. Height information was used to place the ground (TIN surface) and scene features back into 3D space. Top Left: Front view of façade with bays placed in 3D space. Bottom Right: Enlarged view showing car and street sign placement.
Note that only select features are represented. Following the same procedures described in the previous section, flat surfaces that form the balconies and other smooth-surfaces not classified as particular features, such as streetlamps, parking meters, benches, and tree planters can also be represented in the GIS environment. Furthermore, using GIS overlay operations, the classification information can be added to the raw point data.

4.10 Discussion

The research presented here describes the use of object-oriented image processing of urban MTL data. An initial investigation (McQuat et al. 2010) examined the top-down perspective for processing the MTL data and demonstrated that object-oriented analysis was a powerful method for feature extraction in urban environments. However, the top-down perspective did not capture the detail of vertical surfaces and the raster data products were unnecessarily large because they included large areas of un-sampled space (i.e., null values). This project investigated similar methods but from within the scene and taking the sensor perspective as a complementary approach. In this section the methods for working with MTL data in a sensor-perspective coordinate space development of procedural object-oriented feature recognizers are discussed.

4.10.1 Sensor-Perspective Coordinate Space

The interpolation of a set of raster datasets in a scanner-relative space provided a representation of the point cloud that was: a) free of occlusion effects, thus displaying all information for analysis; and b) contained a set of spatial reference raster layers (i.e., layers for each coordinate) so that metrics could be derived despite distortions caused by the sensor-perspective transformation.
A sensor-perspective coordinate space is suitable for establishing the scene model using rules for general feature extraction. Urban geometry relative to the scanner trajectory through the scene is indeed regular. However, working in a non-standard coordinate space made it necessary to define custom metrics for almost all aspects under consideration. While workarounds are possible, many of the potentially useful features of the tool, such as slope and aspect calculation were unusable.

Overall, defining aspects of the urban scene relative to the scanner position provides an excellent framework for MTL feature extraction. The height of the sensor provided a good approximate reference for the ground in the scene and assumptions about how far away building façades are found provided a key spatial division parameter to distinguish building walls from large vehicles such as a bus. Regardless, there remain a number of challenges and limitations to implementing the process described above as illustrated below.

4.10.2 Rule-Based Workflows

Steps in a rule-based workflow are designed to capture regularities in the urban scene. In a procedural workflow the success each step is ultimately limited by the correctness of preceding steps and departures from regularity, as capture by the rule set, can give negative results. As shown in Figure 4-26, an historic church is set back on a corner lot that contains two large trees. The tree trunks are erroneously classified as building façade segments and subsequently, the space between the trees extending away from the scanner towards the church is classified as an architectural bay.
Figure 4-26. Despite the fact that the church does not fit the expected urban geometry along this street scene and the occlusion from large trees on the lot, the recognizers performed well. However, one tree is erroneously classified as building façade, which subsequently led to improperly classified Bay regions.

This example demonstrates two drawbacks to the rule-based workflow. Firstly, it is very difficult (or impossible) to design rules that capture the exact parameter space of a single class. This is exacerbated by imperfect sampling of surfaces due to occlusion as well as within-class variance. Secondly, errors are carried forward through the workflow. Adding new rules and processes to control for these errors as they are found quickly renders workflows overly complex. Despite this, the recognizers still performed reasonably well in most contexts. Ultimately, such an approach may allow automation of areas with regularities. For example, subdivision may exhibit a higher degree of regularity where tuning rules to solve one area will solve the problem for a large number of similar areas. This represents a huge benefit to cities wishing to automate survey practices and to update geospatial databases.

4.10.3 Segmentation

Segmentation of the image into meaningful and representative image-objects is crucial to the object-oriented workflow. Most literature considers multi-resolution segmentation to be part
and parcel of the object-oriented image analysis process. However, after extensive testing, multi-resolution segmentation and other advanced algorithms for segmenting the image did not perform well. Zones of transition and the impetus on maximizing overall heterogeneity within image-objects resulted in segments that did not successfully capture the important boundaries of features in the urban scene.

Initial phases to the segmentation process took the form of identifying surface orientations so that the domain of subsequent phases was constrained and manageable. Entire sections of the workflow were devoted towards identifying zones of transition, which had the effect of improving boundary conditions.

### 4.10.4 Unclassified Image-Objects

Invariably, areas of the image remain unclassified. Some objects are simply too small compared to the number of points returned (Figure 4-27). In many cases, the contiguity was not retained so the feature remains fragmented. Interestingly, a person can still identify many of the unclassified objects.

![Figure 4-27. Low sampling density of small features makes them unidentifiable. However, a human can still easily recognize the characteristic placement and shape of a parking meter.](image)

Returning to the theme of the previous chapter, this implies that the ideal tool will combine semi-automated tools for easily classified features but maintain an intuitive interface for
human recognition of cases that are beyond the scope of the urban scene ontology, do not fit well into a system of rules, or are not captured well at the level of detail captured by the MTL system.

4.10.5 Data Resolution and Rules

A treatment of the fundamental relationship between the scan point density, feature size, and object-oriented processing rules is lacking. Lower sampling densities hide details in surfaces such as cracks and separations that may be important to capture. MTL scans are complex in that point density drops off with distance from the sensor, with changes in beam-surface geometry, and with variable acquisition speed. In addition to developing rules to capture urban regularities, careful study of rules at appropriate resolution is needed. While this was not the goal here, it is nonetheless an important consideration to make if these methods are to be moved into production and thereby represents an essential extension of this project.

4.11 Conclusion

This chapter demonstrates the practicality of using logical rules and the prospect of knowledge-driven feature extraction processes for urban scenes. A scene model captures the location of features in the scene relative to the sensor and relative to other features. It is a promising methodology for urban MTL mapping.

Methods for transforming MTL data into a suitable format for analysis have been presented. Importantly, interpolation of georeferenced position layers (X, Y, Z) and use of such information in the transformed image space was described. This approach demonstrates how such a framework may be used to complement feature extraction from the traditional top-down
perspective and from full 3D environments. Consequently, the approach presented here can potentially be integrated into existing GIS software and feature extraction workflows.

The advantages of the object-oriented tool used in this work result from the design of the procedural workflow, i.e., the capability to define hierarchies of image-objects and rules that guide segmentation and classification. Notably, segmentation did not rely on the specialized segmentation algorithms normally cited as central to the object-oriented process. Moreover, it seems possible to perform similar operations in GIS software where a wider variety of data structures and information can be used – albeit, the workflow would be much more complex and difficult to build.

Designing an object-oriented workflow is largely heuristic. While a general strategy was envisioned, the nuances of the processes, thresholds, and ordering of operations were continually modified. Rules were developed, results examined, and then subsequently refined to remove obvious negative outcomes. It is difficult to account for all the nuances of the process or to foresee all possible consequences of a segmentation or rule-decision. Depending on the scale of analysis, this can mean adapting an overall extraction process to work in an entirely different city from the one for which it was originally developed or changes to a subset of procedures can be tuned for an individual city district, block, or architectural style.

4.12 Future Work

Many of the difficulties in developing the object-oriented workflow presented here have to do with the need to develop methods to work around the non-standard coordinate system and image artifacts, such as transition zones that resulted from interpolation. Future work should be directed towards methods for mitigating these effects, possibly by introducing enhanced pre-
processing steps and should also include a formal treatment of the relationship between rules and scan resolution.

The semantic granularity of the scene model remained at a high level. It is possible to take the analysis of the features/image-objects even further. For example, vehicles can be decomposed into roof, hood, body, wheels, and even reflectors and license plates (using intensity), façade surfaces/objects can also be decomposed to look for more subtle architectural features or have their texture mapped to certain materials, and unique scene-specific features such as tree planters can be included in the model.

Finally, with abstracted features from the raw point cloud now represented in a GIS environment, an urban scene model might take the form of specifying topological rules that can be used to render the model complete. For example, bay polygons and wall lines were produced in the GIS environment but there are obvious undershoot and overshoot cases; topological rules can correct for these and align the bay polygons to the façade lines for a more topologically correct representation.
Chapter 5: Discussion and Future Work

5.1 Discussion

The research presented here addresses various LiDAR user communities, including urban planners and city GIS specialists looking to leverage LiDAR for planning, mapping, and assessment purposes. These communities see the value of new LiDAR technology such as MTL but are faced with the daunting task of extracting value from these data. Workflows and specific methods for extracting features from MTL data were examined, including new visualization techniques for displaying MTL datasets and use of object-oriented software for developing feature extraction methodologies. These methods/applications could be achieved by transforming raw MTL data into a sensor-perspective coordinate space, i.e., displaying 3D data in 2D perspectives suitable for analysis. Ultimately, the goal is to support efficient semi-automated feature extraction into a GIS or CAD environment by end-users.

While an MTL point cloud from an entire survey is considered to be ‘truly 3D’, data from a single line scanner on a MTL system can be represented in two dimensions: 1) position along the acquisition trajectory and 2) at an angle relative to the scanner when a return was logged. However, most GIS, remote sensing and LiDAR processing software are ill-suited for both the visualization and processing of MTL data. Software products typically assume a top-down, 2.5D
perspective on the data that essentially rejects much of the data collected from vertical surfaces. Software for working with static LiDAR scans does not have this limitation.

In Chapter 2, the application of LiDAR technology for mapping cities was introduced. In addition to scanning horizontal features at high resolution (e.g., roads), MTL also captures vertical structure along transportation corridors. It was noted that this aspect of MTL survey – the additional information from features along vertical surfaces at the sub-block scale – had yet to be fully explored prior to this project. For this research, the TITAN® MTL system was described in detail, focusing on its innovative combination of a mobile within-scene point-of-view and the advantages and challenges compared to airborne and static-terrestrial LiDAR scanning practice.

The major challenge to adopting MTL as a city mapping technology is efficiently (i.e. not manually) segmenting the point cloud into sets that represent features of interest such as building façades, signs, and curbs. Difficulty in doing this arises from the large volumes of point data and the complexity of the mobile scanning perspective vis-à-vis urban features. Of these, the highly variable scanner-feature geometry makes applying standard visualization and data processing approaches cumbersome. The issues of geometry, feature sampling, occlusion, and visual complexity related to point clouds from MTL are all constraints on the design of new approaches.

New geovisualization techniques for reducing complexity of the display and user navigation of the point cloud were developed (Chapter 3). The focus was on the user, augmenting the point cloud with additional information so as to clearly relate the connection between data acquisition and resulting point distribution. It was presumed that the trajectory of the MTL system as it moved along the urban corridor and the sensor-perspective of the urban scene were useful perspectives for the user to assess the point coverage and structural aspects of the LiDAR scan, given innate human understanding of urban structure at the street scale. A virtual
environment for viewing MTL data was designed using the Processing® programming language and environment to investigate. The virtual environment – MTL Viz – uses a scanner-centric frame of reference rather than the traditional top-down world reference system common in GIS and RS software or the 3D (X,Y, Z) environments common in virtual survey and 3D game development environments. The data were projected relative to the scanner position from which they were acquired. Moving along the trajectory, the amount of data displayed was adjusted relative to the analysts’ current position. Moreover, a set of novel visualization techniques for augmenting the point cloud display to the user was presented. The techniques focused on communicating information about volumes of scanned urban space – proving some of it to be open and large portions to be occluded from scanner view. Of key importance was that the visualization techniques made this characteristic of the MTL data obvious and intuitive to the user.

In addition to illustrating the LiDAR beam path between the sensor and a point in the scene, the results showed the space that was scanned (or not scanned). This offers a significant advantage for interpretation of scanned urban structures (i.e., sensor-perspective on the scene made visualizing these aspects possible). It was noted that such a technique might be used for quality control and assessment. It may also be integrated into existing manual or semi-automated feature extraction workflows. Visualization techniques that augmented the view were important to communicate scanner operation and the direction that the system was travelling relative to the urban corridor. This investigation into use of the scanner perspective for visualization and user interaction with MTL data was performed as a first step towards creating more usable visualization and analysis tools. One significant element that, while noted, was not pursued was the explicit analysis of occlusion and “un-scanned space”.

106
Automated methods play a supportive role in the feature extraction process. It was noted that automated segmentation of LiDAR points from aerial and static scanner data take advantage of scanner perspectives on scanned features. The scanner perspective appears to be key to segmentation of MTL data as well (Zhao & Shibasaki 2003, Fruh et al. 2005, Carlberg et al. 2009). However, the semantic granularity of segment classification remains general. Knowledge-based methods use contextual knowledge to drive segmentation routines and are used to improve classification granularity by incorporating general knowledge about the inter-relationships between features in an urban scene.

Methods for applying object-oriented image-analysis to improve the semantic granularity of urban MTL classification were developed in Chapter 4. The object-oriented process was designed around the principle of a scene-model where logical sets of rules captured the relationships between scene features such as ground, building façade, and vehicles. As with the visualization tool described in Chapter 3, the MTL scanner perspective on the urban scene was exploited and was found to offer advantages by effectively removing occluded regions from the analysis while displaying data along both flat and vertical surfaces. MTL data, transformed into scanner-perspective coordinate space, were interpolated into a continuous along-track raster image of the scene for processing in Definiens’ eCognition®. The X, Y, and Z coordinates of the original data were maintained as respective raster layers. Segmentation of the transformed MTL data was performed using local patch-growing procedures. The general strategy was to segment and classify the largest structures in the scene first (i.e., ground and façade), which established the context then used to construct more detailed logical rules to recognize smaller features (e.g., cars and signs) at subsequent stages.
Overall, the design process of automated feature extraction using object-oriented software was largely heuristic, despite the seemingly simple rules that govern individual feature recognition. Heuristic investigation was needed to remove anomalous segmentation artifacts or to determine conditions to prevent them from occurring. The advantage of using Definiens’ eCognition® software was due to the integration of raster and vector formats: i.e., the ability to query image-object properties; to easily construct procedures out of rules and algorithms; and access hierarchical and neighbourhood relationships between image-objects.

Overall, this project contributes to our understanding of methods for urban analysis using LiDAR data. Key outcomes are:

1) Evaluation of a within-scene perspective for visualization and processing of MTL data using scan-angle and along-track distance. This technique hides artifacts related to the scan geometry, clearly displaying MTL data to the user and providing a suitable surrogate image for processing 3D data in 2D.

2) Suggesting and investigating a focus on the user of MTL data for visualization and for the development of semi-automated feature extraction procedures.

3) The concept of proven open space from LiDAR survey. Its application was illustrated by extracting volumes inside of buildings as seen through windows.

4) An urban scene model consisting of a logical rule set and procedural semi-automated feature extraction using emerging object-oriented image processing technology.

5) A fluid transfer of features back into CAD / GIS.

The limitations are:

a) The semi-automatic feature extraction procedures are sensitive to level of detail (i.e. precision, scan coverage, and semantics) and error.

b) Non-standard architecture is not captured well by logical rule sets resulting in artifacts.
c) Some features would be much easier to recognize in true 3D. Working in 2D means that several stages are needed for different 3D perspective views, e.g., the many facets of a car body need to be recognized separately and merged together using a complex set of rules and procedures.

d) The current feature extraction system is not “production ready”.

5.2 Future Work

This research explored the potential for visualizing and analyzing MTL data in a trajectory / scanner-perspective coordinate space. That is, the trajectory of the MTL system through an urban scene and the angular resolution of the scanner were used to represent MTL data from the sensor perspective. Methods for working with data from this perspective show promising avenues for further research as do addressing some of the limitations outlined above. Future work will focus on analysis of scanned versus un-scanned space in the urban scene, improved GIS-based processing of MTL data, workflow analysis of MTL Viz-type tools, and refinement of the scene model concept towards a formal ontology with respect to MTL and its role in urban GIS.

Aspects concerning mapping what has and has not been scanned are important to practitioners. Topics of concern are the development metrics that capture the degree to which space is proven to be open by complement of a return, e.g., sampling is less dense as distance from the scanner increases. This has implications for what level of detail can be represented in what regions of the sampled urban scene. Visualization tools are a first step towards attempting to understand this and the development of methods to map this will be of interest as well.

Furthermore, there are opportunities for application of existing tools in concert with the development of new tools for processing MTL data within a GIS environment. Data can be
segmented initially into foreground and background objects using preprocessing steps as shown by Fruh et al. (2005). Gaps in background points, e.g., belonging to façade, could be 'filled' to provide better 'surface' models of the urban fabric and make recognition tasks simpler for foreground objects. Furthermore, there exists the opportunity to develop a toolset capable of mapping between the sensor perspective outlined here and common geographic projections used in GIS. Such a tool would enable the full application of the GIS suite of tools and algorithms, opening new avenues for MTL analysis. GIS-integrated MTL tools might look something like MTL Viz (Chapter 3). The sensor-perspective display and visualization techniques need to be evaluated within a production environment and compared to existing tools and workflows.

Lastly, ontology of urban scene models at the scales commensurate with MTL and static-terrestrial LiDAR should be defined formally. As GIS becomes increasingly web-centric and matures to 3D and 4D representation and analyses, strong ontologies are going to be ever more important and the idea of an MTL scene-model will be useful for integrating data into new GIS paradigms. Therefore, in addition to GIS-integration of MTL-specific tools, refinement of the scene-model concept represents an exciting avenue for future work.
References


111


Lato, M.J. 2010. Geotechnical applications of LiDAR pertaining to geomechanical evaluation and hazard identification, *Ph. D. Dissertation*, Queen’s University, NR65113.


Appendix A

MTL Viz – Processing® Code

// MTL Viz
// G.McQuat 2010

// The data input expected for this program consists
// of an ascii text containing at least point and sensor locations
// for each point.

// Globals
String asciipts[];
float[][] pts;
float[][] pts2;
float measured_slice_size;
float measured_slice_location;
int measured_lower_offset;
int measured_upper_offset;
int location;
int offset;
int forwardStep;
int backStep;
boolean clearsreen = false;
boolean change = true;
boolean newShape = true;
boolean drawEmptyPolys = false;
boolean angleView = true;
boolean measure = false;
PFont aFont;
int datalength;
int sensor_x=500;
int sensor_y=250;
int u_scale;
int draggable;
int image3d_x=750;
int image3d_y=250;

Slider offset_slider;
Slider location_slider;
Slider step_slider;
Button stepup;
Button stepdown;
Button emptyPolys;
Button draw3D;
Button measure_button;

MeasureTool measure_tool;

//Initialize the application
void setup() {

//Screen
size(1024, 700);
background(255);
noLoop();

//User Along-Track Variables
offset=10000;
location = offset;
forwardStep=200;
sensor_x = 500;
sensor_y = 250;
u_scale = 20;

//Offsets
measured_lower_offset = 0;
measured_upper_offset = 0;

//Screen Text
aFont = createFont("Calibri", 10);
textFont(aFont);

//Load point data from text file
//For now we have to add a line for each file we use.
//In a full version it would need to allow the user to select many files.
//For particular systems the angle relative to the system travel would also need to be given
//by the user and this is beyond the project scope.

//Some feedback
println(str(minute()) + ":" + str(second()) + " Loading Data...");
asciipts = loadStrings("scan2pts.ascii");

dataLength = asciipts.length / 6; //Expecting both point and sensor locations - IN ORDER!
pts = new float[datalength][7];
pts2 = new float[datalength][2]; //Two extra fields to store offsets from sensor.
println(datalength +" Points.");

//Populate the double-precision matrix
println(str(minute()) + ":" + str(second()) + " ASCII --> Float...);
int j=0;
for (int i=0; i < (datalength*6); i=i+1) {
    pts[i/6][j] = float(asciipts[i]);
    j=j+1;
    if(j==6) {
        j=0;
    }
}

//Calculate Along-Track Distances in the 7th table column.
println(str(minute()) + ":" + str(second()) + " Along-Track Distance Calculations...");
pts[0][6] = 0;
for (int i=1; i < datalength; i=i+1) {
    pts[i][6] = pts[i-1][6] + sqrt(pow(pts[i-1][3]-pts[i][3],2) + pow(pts[i-1][4]-pts[i][4],2));
}
println(str(minute()) + ":" + str(second()) + " Calculating Point-Sensor Offsets...");
for (int i=0; i < datalength; i=i+1) {
    pts2[i][0] = dist(pts[i][0],pts[i][1],pts[i][3],pts[i][4]); //xy Distance between sensor and point.
    pts2[i][1] = pts[i][5]-pts[i][2];
}
println(str(minute()) + ":" + str(second()) + " Setup GUI Objects...");

//Sliders
offset_slider = new Slider(99,21,0,datalength/8,12,400);
location_slider = new Slider(99,9,0,datalength,12,400);
offset_slider.currentX = (offset/int(offset_slider.stepSize))+offset_slider.posX;

//Buttons
stepup = new Button(99,35,12,12);
stepdown = new Button(99,48,12,12);
emptyPolys = new Button(99,60,20,12);
draw3D = new Button (99,74,20,12);
measure_button = new Button (99,86,20,12);

//Measure tool
measure_tool = new MeasureTool();
println("Done Setup");
}

//////////////////////////////////////////////// DRAW //////////////////////////////////////

void draw() {
    select_measured_slice(measured_slice_location,measured_slice_size);
    draw_time_slice(offset);

    //Mouse Location
    if (mousePressed == true) {
        loop();
        clearscreen = true;
    }

    // The measure tool
    if (measure == true) {
        measure_tool.show();
    }

    //Draw text, sliders, buttons //////////

    //Sliders
    offset_slider.show();
    location_slider.show();

    //Buttons and Boxes
    stepup.show();
    stepdown.show();
    emptyPolys.show();
    measure_button.show();

    //Text
    fill(0,0,0);
    text("Current Point: ", 10, 20);
    text(location, 100, 20);
    text("Points Displayed: ", 10, 32);
    text(offset, 100, 32);
    text(forwardStep + " points",stepdown.posX+stepdown.bWidth+5,stepdown.posY);
    text("Step by ",10,stepdown.posY);
    text(" ^",stepup.posX,stepup.posY+stepup.bHeight*0.75);
    text(" v",stepdown.posX,stepdown.posY+stepdown.bHeight*0.75);
    text(str(drawEmptyPolys),emptyPolys.posX,emptyPolys.posY+emptyPolys.bHeight-1);
    text("Open Space: ",10,72);
    text("Measure Tool: ",10,92);
}
//Select a measured slice along the trajectory path.
void select_measured_slice(float slice_location, float slice_size) {
    int measured_search;

    //Start the search from the current location in the array.
    //Find a point < 10cm from the slice location
    if (slice_location < pts[location][6]) {
        measured_search = -1;
    }
    else {
        measured_search = 1;
    }

    //Now find the lower offset for the measured_slice_size
    while(abs(pts[location][6]-pts[measured_lower_offset][6]) < slice_size) {
        measured_lower_offset = measured_lower_offset - 1;
    }

    //And the upper offset for the measured_slice_size
    while(abs(pts[location][6]-pts[measured_upper_offset][6]) < slice_size) {
        measured_upper_offset = measured_upper_offset + 1;
    }

    //Select a time slice from the array and show it
    void draw_time_slice(int offset) {

        //Locals
        float q;
        float qmax;
        int k;

        //Allocate the appropriate amount of memory.
        //double time_slice[] = subset(datapts slice_size);
        //Select it from the dataset.
        //Draw a certain subset of the points

        //Additional Boundary tests
        if (location < 0) {
            location = offset + 2;
        }

        if (location-offset < 0) {
            offset = location - 2;  
        }
if (location >= datalength) {
    location = datalength - 1;
}

if (clearscreen == true) {
    background(255);
}

float transparency_step = 255 / float(offset);
float transparency = 0;

//////////////////////////////// Triangles //////////////////////////////////

if (drawEmptyPolys == true) {
    k = location;
    transparency_step = 10;
    transparency = 200;
    noStroke();
    fill(0,255,0,transparency);
    for(int i = 3; i > 0 & & k>0; i = i - 1) {
        beginShape();
        transparency = transparency + transparency_step;
        vertex(sensor_x,sensor_y);
        vertex(sensor_x - (u_scale * pts2[k][0]), sensor_y + (u_scale * pts2[k][1]));
        while (k>0 & & atan(pts2[k-1][1]/pts2[k-1][0]) - atan(pts2[k][1]/pts2[k][0]) < 0.05) {
            (u_scale * pts2[i-1][0]), (sensor_y + (u_scale * pts2[i][1]));
            vertex(sensor_x - (u_scale * pts2[k-1][0]), sensor_y + (u_scale * pts2[k-1][1]));
            k = k - 1;
        }

        //Draw the empty space at the top of the arc, accounting for system roll.
        if(k>0){
            qmax = atan(pts2[k-1][1]/pts2[k-1][0]) - 1.4;
            q = atan(pts2[k+1][1]/pts2[k+1][0]);
            while (q > qmax) {
                vertex(sensor_x - (u_scale * 75 * cos(q)), sensor_y + (u_scale * 75 * sin(q)));
                q = q - 0.007;
            }
        }
        k=k-1;
    endShape();
}
// Points Around the current location.
transparency = 0;
stroke(0,0,1,transparency);

// Currently draw equal amounts on either side of current location.
for (int i = int(location-(offset/2)); i < int(location+(offset/2)) && i < datalength; i = i+1) {
    transparency = transparency+transparency_step;
stroke(0,0,255,transparency);
    point((sensor_x - (u_scale * pts2[i][0])), (sensor_y + (u_scale * pts2[i][1])));
}

////////////////////////////////// Across-Track Perspective ///////////////////////////////////
// Want to be able to draw geoide-relative height, sensor-relative height, or polar angle on Y-axis.
if(angleView == true) {
    transparency = 0;
k = 0;
beginShape(POINTS);
for (int i = int(location-offset+1); i < int(location+offset) && i < datalength-2; i = i+1) {
    while (atan(pts2[i-1][1]/pts2[i-1][0]) - atan(pts2[i][1]/pts2[i][0]) < 0.05 && i < (datalength-1)) {

        // Point colouring at 15m intervals RGB.
        stroke(dist(pts2[i][0],pts2[i][1],pts2[i+1][0],pts2[i+1][1])*100,0,0);
        vertex(sensor_x + 50 + k, sensor_y + u_scale*pts2[i][1]);
i++;
    }
    k++;
}
endShape();

fill(255,0,0,100);
ellipse(sensor_x + 50 + k/2, sensor_y + u_scale*pts2[location][1],10,10);
paint(sensor_x + 50 + k/2, sensor_y + u_scale*pts2[location][1]);

}
rect(sensor_x-5,sensor_y-5,10,10);
ellipse(sensor_x - (u_scale * pts2[int(location)][0]),sensor_y + (u_scale * pts2[int(location)][1]),2,2);
clearscreen = false;
change = true;
noLoop();
}

//Scrolling forwards and backwards.
void keyPressed() {
  if (key == CODED) {
    if (keyCode == UP) {
      location = location + forwardStep;
      measured_slice_location = measured_slice_location + 1;
      if (location > datalength){
        location = datalength-2;
      }
    }
    if (keyCode == DOWN) {
      location = location - forwardStep;
      measured_slice_location = measured_slice_location - 1;
      if (location < 0){
        location = 2;
      }
    }
  }
  if (key == 's') {
    u_scale= u_scale + 2;
  }
  if (key == 'a') {
    u_scale = u_scale - 2;
  }
  clearscreen = true;
  loop();
}

void mousePressed() {
  clearscreen = true;
  loop();

  // Initiate the measure tool
  if (measure_tool.started == false && measure == true) {
    measure_tool.go();
else if (forwardStep == 1) {
    forwardStep = forwardStep + 1;
}

////////////  Change the step speed  /////////////
if (mouseX < stepup.posX+stepup.bWidth && mouseX > stepup.posX && mouseY <
    stepup.posY+stepup.bHeight && mouseY > stepup.posY && forwardStep < 10) {
    forwardStep = forwardStep + 1;
}
else if (mouseX < stepup.posX+stepup.bWidth && mouseX > stepup.posX && mouseY <
    stepup.posY+stepup.bHeight && mouseY > stepup.posY && forwardStep < 100) {
    forwardStep = forwardStep + 10;
}
else if (mouseX < stepup.posX+stepup.bWidth && mouseX > stepup.posX && mouseY <
    stepup.posY+stepup.bHeight && mouseY > stepup.posY && forwardStep < 1000) {
    forwardStep = forwardStep + 100;
}
else if (mouseX < stepup.posX+stepup.bWidth && mouseX > stepup.posX && mouseY <
    stepup.posY+stepup.bHeight && mouseY > stepup.posY && forwardStep < 10000) {
    forwardStep = forwardStep + 1000;
}

//  going down...
if (mouseX < stepdown.posX+stepdown.bWidth && mouseX > stepdown.posX && mouseY <
    stepdown.posY+stepdown.bHeight && mouseY > stepdown.posY && forwardStep > 1 &&
    forwardStep <= 10) {
    forwardStep = forwardStep - 1;
}
else if (mouseX < stepdown.posX+stepdown.bWidth && mouseX > stepdown.posX && mouseY <
    stepdown.posY+stepdown.bHeight && mouseY > stepdown.posY && forwardStep > 1 &&
    forwardStep <= 100) {
    forwardStep = forwardStep - 10;
}
else if (mouseX < stepdown.posX+stepdown.bWidth && mouseX > stepdown.posX && mouseY <
    stepdown.posY+stepdown.bHeight && mouseY > stepdown.posY && forwardStep > 1 &&
    forwardStep <= 1000) {
    forwardStep = forwardStep - 100;
}
else if (mouseX < stepdown.posX+stepdown.bWidth && mouseX > stepdown.posX && mouseY <
    stepdown.posY+stepdown.bHeight && mouseY > stepdown.posY && forwardStep > 1 &&
    forwardStep <= 10000) {
    forwardStep = forwardStep - 1000;
}

///////////  The Open Space Polygon Button  /////////////
if (mouseX < emptyPolys.posX+emptyPolys.bWidth && mouseX > emptyPolys.posX &&
mouseY < emptyPolys.posY+emptyPolys.bHeight && mouseY > emptyPolys.posY) {
  if (drawEmptyPolys === true) {
    drawEmptyPolys = false;
  }
  else {
    drawEmptyPolys = true;
  }
}

/////////// The Measure Button /////////////
if (mouseX < measure_button.posX+measure_button.bWidth && mouseX >
measure_button.posX && mouseY < measure_button.posY+measure_button.bHeight &&
mouseY > measure_button.posY) {
  if (measure == true) {
    measure = false;
  }
  else {
    measure = true;
  }
}

//When over the offset slider bar
if(mouseX > offset_slider.posX && mouseX < offset_slider.posX+offset_slider.sLength &&
mouseY > offset_slider.posY && mouseY < offset_slider.posY+offset_slider.sSize){
  offset_slider.currentX=mouseX;
  offset = (offset_slider.currentX-offset_slider.posX)*int(offset_slider.stepSize);
}

//When over the location slider bar
if(mouseX > location_slider.posX && mouseX <
location_slider.currentX+location_slider.sLength && mouseY > location_slider.posY &&
mouseY < location_slider.posY+location_slider.sSize){
  location_slider.currentX=mouseX;
  location = (location_slider.currentX-location_slider.posX)*int(location_slider.stepSize);
}

// When the mouse is pressed and dragged.
void mouseDragged() {

  //If selecting the sensor box
  if (mouseX > sensor_x-5-draggable && mouseX < sensor_x+5+draggable && mouseY >
sensor_y-5-draggable && mouseY < sensor_y+5+draggable){

draggable=50;
sensor_x=mouseX;
sensor_y=mouseY;
}

//When over the 3D sensor box
if(mouseX > image3d_x-draggable-5 && mouseX < image3d_x+draggable+5 && mouseY > image3d_y-draggable-5 && mouseY < image3d_y+5+draggable){
  draggable=50;
  image3d_x=mouseX;
  image3d_y=mouseY;
}

//When over the offset slider bar
if(mouseX > offset_slider.posX && mouseX < offset_slider.posX+offset_slider.sLength && mouseY > offset_slider.posY && mouseY < offset_slider.posY+offset_slider.sSize){
  offset_slider.currentX=mouseX;
  offset = (offset_slider.currentX-offset_slider.posX)*int(offset_slider.stepSize);
}

//When over the location slider bar
if(mouseX > location_slider.posX && mouseX < location_slider.currentX+location_slider.sLength && mouseY > location_slider.posY && mouseY < location_slider.posY+location_slider.sSize){
  location_slider.currentX=mouseX;
  location = (location_slider.currentX-location_slider.posX)*int(location_slider.stepSize);
}

clearscreen = true;
loop();
}

void mouseReleased () {
  draggable=0;
}

void mouseMoved(){
clearscreen = true;
loop();

//Cursor
if (measure == true) {
cursor(CROSS);
}
else {
cursor(ARROW);
}

//Do things if over an editable object.
if (mouseX > 100 && mouseX < 160 && mouseY > 9 && mouseY < 30){
cursor(TEXT);
}

if (mouseX > sensor_x-5 && mouseX < sensor_x+5 && mouseY > sensor_y-5 && mouseY < sensor_y+5){
cursor(HAND);
}

if(mouseX > image3d_x-5 && mouseX < image3d_x+5 && mouseY > image3d_y-5 && mouseY < image3d_y+5){
cursor(HAND);
}

//When over the offset slider
if(mouseX > offset_slider.currentX-draggable && mouseX < offset_slider.currentX+draggable+offset_slider.sSize && mouseY > offset_slider.posY && mouseY < offset_slider.posY+offset_slider.sSize+draggable){
cursor(HAND);
}

//When over the location slider
if(mouseX > location_slider.currentX-draggable && mouseX < location_slider.currentX+draggable+location_slider.sSize && mouseY > location_slider.posY && mouseY < location_slider.posY+location_slider.sSize+draggable){
cursor(HAND);
}

// SLIDERS
class Slider {
    //Class Variables
    int posX;
    int posY;
    int startVal;
    int endVal;
    int sSize;
    int sLength;
    int currentX;
    float stepSize;
//Slider Constructor Method
Slider(int tempposX, int tempposY, int tempstartVal, int tempendVal, int tempsSize, int tempsLength) {

//Assigns the given values to the objects' variables
posX = tempposX;
posY = tempposY;
currentX = 10;
startVal = tempstartVal;
endVal = tempendVal;
sSize = tempsSize;
sLength = tempsLength;
stepSize = (endVal - startVal)/sLength;
}

void show () {
    if (mouseX < currentX + sSize && mouseX > currentX && mouseY < posY + sSize && mouseY > posY) {
        fill(0, 255, 0);
    } else {
        fill(0, 255, 0, 50);
    }
    stroke(255, 0, 0);
    rect(currentX, posY, sSize, sSize);
    fill(0, 255, 0, 50);
    rect(posX, posY, sLength, sSize);
}
}

///// BUTTONS /////////////////
class Button {

//Button Variables
int posX;
int posY;
int bWidth;
int bHeight;

//Button Constructor
Button (int tposX, int tposY, int tbwidth, int tbHeight) {
posX = tposX;
posY = tposY;
bWidth = tbwidth;
bHeight = tbHeight;
}

//Methods for Button
void show() {
    if (mouseX < posX+bWidth && mouseX > posX && mouseY < posY+bHeight && mouseY > posY) {
        fill(0,255,0);
    }
    else {
        fill(0,255,0,50);
    }
    rect(posX,posY,bWidth,bHeight);
}

// Measure Tool /////////////////////////////////////////////////
class MeasureTool {
    //Variable
    boolean started;
    float startX;
    float startY;
    float endX;
    float endY;

    //Constructor for Measure Tool
    MeasureTool() {
        started = false;
    }

    //Methods for Measure Tool
    void go() {
        started = true;
        startX = mouseX;
        startY = mouseY;
    }

    void stop() {
        started = false;
        endX = mouseX;
        endY = mouseY;
    }

    void show() {
        strokeWeight(2);
    }
}
if (started == false) {
  line(startX, startY, endX, endY);
  text(str(dist(startX, startY, endX, endY)/u_scale) + "m", endX+10, endY+10);
}
else {
  line(startX, startY, mouseX, mouseY);
  text(str(dist(startX, startY, mouseX, mouseY)/u_scale), mouseX+10, mouseY+10);
  text("Range: " +
        str(dist(mouseX, mouseY, sensor_x, sensor_y)/u_scale), mouseX+10, mouseY+25);
  strokeWeight(1);
}
Appendix B

Class Descriptions

Automobile Body
or (max)
and (min)
[0.2-1.5]: Height From Ground
Threshold: side-look range < 8
[2.99-6]: XY Vector
and (min)
Threshold: XY Vector >= 0.5
[0.1-0.5]: Height From Ground
Threshold: Max. pixel value Intensity > 120

Bay
and (min)
[0-0.2]: Mean. Diff. to N. Facade side-look Range

Car
and (min)
Threshold: Classified as Automobile Body = 1
Threshold: Height <= 1.5
connected-to ground
connected-to side-wall

Curb:
and (min)
[0-0.2]: XY Vector
[0.05-0.27]: Height
[0-1]: Rel. border to Surface

Disconnected
and (min)
Threshold: Area < 200 Pxl
Threshold: Mean Diff. Max Range to Facade > 0.1
Threshold: Mean. Diff. to N. Facade side-look Range <= -0.5

Facade Segment
and (min)
Threshold: side-look range > 8
Threshold: Height >= 2.5
Threshold: XY Vector >= 1

Flat Architectural Component
and (min)
Threshold: Mean HFS > 0
Threshold: Existence of sub objects Smooth Flat (1) = 1
Threshold: Area (Depth * ATD Range) >= 5 Pxl
Threshold: Border to Facade Segment > 0 Pxl

**outlier_bay**
**outlier_projection**
**Outlier**

**Projection**
and (min)
  Threshold: Mean diff. to Range, Facade Segment > 0

**Roof**
and (min)
  Threshold: Vertical Area (Height * XY Vector) >= 3
  Threshold: Height From Ground > 3

**Side Wall**
and (min)
  Threshold: Vertical Area (Height * Depth) >= 20

**Sign**
and (min)
  Threshold: XY Vector > 0.1
  [1.5-3]: Height From Ground
  Threshold: Mean Intensity >= 120

**Smooth Flat**
and (min)
  [-0.015-0.015]: Abs. Mean Diff. to UC Z(0)

**Smooth Side**
and (min)
  Threshold: Depth >= 0.2
  Threshold: Height >= 0.2

**Smooth Sloped**
**Smooth Vertical**
and (min)
  Threshold: ATD Range >= 0.2
  Threshold: Height >= 0.2

**Step:**
and (min)
  Threshold: Height From Ground < 0.5
  [0.03-0.2]: Height
  [0-1]: Rel. border to Surface
  [0-1]: Depth
  Threshold: Mean Range > 8
  Threshold: XY Vector >= 0.9

**Surface**
and (min)
  Threshold: Height From Ground < 0.2
  Threshold: XY AREA (EST.) >= 5
\textit{VanBus} \\
and (min) \\
Threshold: Classified as Automobile Body = 1 \\
Threshold: Height $\geq 2$ \\
\textit{VanTruckSuv} \\
and (min) \\
Threshold: Classified as Automobile Body = 1 \\
Threshold: Height $< 2$ \\
Threshold: Height $> 1.5$
Appendix C

Customized Metrics

1. ATD Dist. From -10deg Line: ([X Center]*0.076)-([Mean Range]* tan (-10))
2. ATD Range: 0.076*([X Max]-[X Min])
3. Area (Depth * ATD Range): [Depth]*[ATD Range]
4. Depth: [Max. pixel value Range]-[Min. pixel value Range]
5. Height: [Max. pixel value Z]-[Min. pixel value Z]
6. Height From Ground: [Mean HFS]+3
7. Slope (arctan Yrange/xRange): arctan ([Y Range]/[X Range])
8. Vertical Area (Height * Depth): [Height]*[Depth]
9. Vertical Area (Height * XY Vector): [Height]*[XY Vector]
10. X Range: [Max. pixel value X]-[Min. pixel value X]
11. XY AREA (EST.): ([Max. pixel value X]-[Min. pixel value X])*([Max. pixel value Y]-[Min. pixel value Y])
12. XY Vector: ([X Range]^2+[Y Range]^2)^0.5
13. Y Range: [Max. pixel value Y]-[Min. pixel value Y]
14. side-look range: [Mean Range]* cos (10)
15. Abs Mean. Diff. to N. Facade side-look Range: 'Mean absolute difference' of 'side-look range' of Facade Segment neighbours [0 Pxl]
16. Abs. Mean Diff Z to Smooth Flat: 'Mean absolute difference' of 'Mean Z' of Smooth Flat neighbours [0 Pxl]
17. Abs. Mean Diff. to UC Z(0): 'Mean absolute difference' of 'Mean Z' of unclassified neighbours [0 Pxl]
18. Abs. Mean. Diff. to N. UC side-look Range: 'Mean absolute difference' of 'side-look range' of unclassified neighbours [0 Pxl]

19. Abs. Mean. Diff. to Smooth Vert. Range: 'Mean absolute difference' of 'side-look range' of Smooth Vertical neighbours [0 Pxl]

20. Diff. ATD to Side (Pxl): 'Mean absolute difference' of 'ATD Dist. From -10deg Line' of Smooth Side neighbours [0 Pxl]

21. Height_Super: 'Mean' of 'Height' of super-object [0]

22. Mean Diff. of Mean Diff ATD to UCN (Pxl): 'Mean absolute difference' of 'Mean Diff. ATD to UCN' of unclassified neighbours [0 Pxl]

23. Mean Diff. ATD to UCN (Pxl): 'Mean absolute difference' of 'ATD Dist. From -10deg Line' of unclassified neighbours [1 Pxl]

24. Mean Diff. Max Range to Facade: 'Mean absolute difference' of 'Max. pixel value Range' of Facade Segment neighbours [0 Pxl]

25. Mean. Diff. to N. Facade side-look Range: 'Mean difference' of 'side-look range' of Facade Segment neighbours [0 Pxl]

26. MeanX_Super Object: 'Mean' of 'Mean X' of super-object [0]

27. MeanY_SuperObject: 'Mean' of 'Mean Y' of super-object [0]

28. MeanZ_SuperObject: 'Mean' of 'Mean Z' of super-object [0]

29. MinZ_Super: 'Min' of 'Min. pixel value Z' of super-object [0]

30. mean diff max range to auto: 'Mean difference' of 'Max. pixel value Range' of Automobile Body neighbours [0 Pxl]
Appendix D

Main Procedure

Urban Scene Feature Extraction and Recognizers

1. Initialize the workspace
   a. [chessboard segmentation: chess board: 1 creating 'Pixel Level']
   b. [create temporary image layer: at Pixel Level: Create temp. image layer 'ATD' using 'ATD Dist. From -10deg Line']

2. Flat Surfaces
   a. merge region: with Mean Diff. to neighbors (abs) Z (1) <= 0.01 at Pixel Level: merge region
   b. assign class: with XY AREA (EST.) >= 0.2 at Pixel Level: Smooth Flat
   c. merge region: Smooth Flat at Pixel Level: merge region
   d. chessboard segmentation: unclassified at Pixel Level: chess board: 1

3. Vertical Surfaces
   a. Front
      i. merge region: unclassified with Abs. Mean. Diff. to N. UC side-look Range <= 0.025 at Pixel Level: merge region
      ii. image object fusion: loop: unclassified with Area > 2 Pxl at Pixel Level: <- unclassified: unclassified + all side-look range <= 0.025
      iii. classification: unclassified at Pixel Level: Facade Segment, Smooth Vertical
   b. Side
      i. merge region: unclassified with Mean Diff. ATD to UCN < 0.03 Pxl at Pixel Level: merge region
      ii. classification: unclassified at Pixel Level: Side Wall, Smooth Side

4. Sloped and Other Smooth
   a. merge region: loop: unclassified with Mean Diff of Mean Diff ATD to UCN < 0.003 Pxl at Pixel Level: merge region
   b. classification: unclassified at Pixel Level: Roof
   c. assign class: loop: unclassified with Rel. border to Roof >= 0.5 at Pixel Level: Roof
   d. merge region: Roof at Pixel Level: merge region
5. **Anneal**
   a. chessboard segmentation: Facade Segment, Smooth Flat, Smooth Side, Smooth Vertical at Pixel Level: chess board: 2
   b. grow region: loop: Facade Segment at Pixel Level: <- unclassified Abs Mean. Diff to N. Facade side-look Range <= 0.05
   c. grow region: loop: Smooth Vertical at Pixel Level: <- unclassified Abs. Mean Diff. to Smooth Vert. Range <= 0.05
   d. grow region: loop: Smooth Side at Pixel Level: <- unclassified Diff. ATD to Side < 0.05 Pxl
   e. grow region: loop: Smooth Flat at Pixel Level: <- unclassified Abs. Mean Diff Z to Smooth Flat <= 0.02

6. **Post Anneal Cleanup**
   a. merge region: Smooth Flat at Pixel Level: merge region
   b. merge region: Smooth Vertical at Pixel Level: merge region
   c. assign class: Smooth Vertical with Abs Mean. Diff. to N. Facade side-look Range <= 0.05 at Pixel Level: Facade Segment
   d. merge region: Facade Segment at Pixel Level: merge region
   e. merge region: Smooth Side at Pixel Level: merge region
   f. classification: Smooth Vertical at Pixel Level: Facade Segment
   g. classification: Smooth Flat at Pixel Level: Surface
   h. Curbs and Steps
   i. chessboard segmentation: Smooth Vertical with Height From Ground < 0.05 at Pixel Level: chess board: 20
      i. assign class: Smooth Vertical with Height From Ground < 0.05 at Pixel Level: unclassified
      ii. classification: unclassified at Pixel Level: Curb, Step
   j. assign class: loop: unclassified with Rel. border to Step >= 0.5 at Pixel Level: Step
      i. merge region: Step at Pixel Level: merge region
   k. assign class: unclassified with XY AREA (EST.) >= 1 at Pixel Level: Smooth Sloped
   l. classification: Smooth Side at Pixel Level: Side Wall

7. **Reflective Signs**
   a. classification: Smooth Side, Smooth Vertical, unclassified at Pixel Level: Sign

8. **Prep for Abstraction**
   a. spectral difference segmentation: unclassified at Pixel Level: spectral difference 0
   b. assign class: loop: unclassified with Abs Mean. Diff. to N. Facade side-look Range < 0.1 at Pixel Level: Facade Segment

   c. **Outliers**
      i. **Facade-Relative**
         1. assign class: with Width = 1 Pxl at Pixel Level: Outlier
2. chessboard segmentation: Outlier at Pixel Level: chess board: 1
3. assign class: Outlier with Mean diff. to Range, Facade Segment < 0 at Pixel Level: outlier_projection
4. assign class: loop: Outlier with Mean diff. to Range, outlier_projection <= 0 at Pixel Level: outlier_projection
5. merge region: outlier_projection at Pixel Level: merge region
6. assign class: Outlier with Mean diff. to Range, Facade Segment > 0 at Pixel Level: outlier_bay
7. assign class: loop: Outlier with Mean diff. to Range, outlier_bay > 0 at Pixel Level: outlier_bay
8. merge region: outlier_bay at Pixel Level: merge region
9. Remove Slivers & Inconsistencies
10. assign class: outlier_projection with Width = 1 Pxl at Pixel Level: Outlier
11. assign class: outlier_bay with Number of outlier_projection (0) >= 2 at Pixel Level: Outlier
12. chessboard segmentation: Outlier at Pixel Level: chess board: 1

e. Create Level 2
   i. copy image object level: at Pixel Level: copy creating 'Level 2' above
   ii. merge region: Facade Segment at Level 2: merge region
   iii. merge region: Side Wall at Level 2: merge region

9. Automobiles
a. classification: Smooth Side, Smooth Vertical at Level 2: Automobile Body
b. assign class: Smooth Flat with Mean Range < 5 and Height From Ground > 0.5 at Level 2: Automobile Body
c. image object fusion: loop: Automobile Body at Level 2: <- Automobile Body, Smooth Flat, Smooth Side, Smooth Vertical, unclassified: Automobile Body + all Mean Range < 1
d. image object fusion: loop: Automobile Body at Level 2: <- Smooth Side: Automobile Body + all Mean ATD < 1
e. assign class: loop: Smooth Flat, Smooth Side, Smooth Vertical, unclassified with Mean diff. to Range, Automobile Body > 0 and mean diff max range to auto <= 0 at Level 2: Automobile Body
f. merge region: Automobile Body at Level 2: merge region
g. classification: Automobile Body at Level 2: Car, VanBus, VanTruckSuv
10. Bays, Projections & Disconnected
   a. Facade-Relative
      i. assign class: Outlier, outlier_bay, outlier_projection, Smooth Side, Smooth Vertical, unclassified with Border to Side Wall > 0 Pxl at Level 2: connected-to side-wall
      ii. assign class: unclassified with Border to Surface > 0 Pxl at Level 2: connected-to ground
      iii. [classification: Outlier, Smooth Flat, Smooth Side, Smooth Sloped, Smooth Vertical, unclassified at Level 2: Disconnected]
      iv. chessboard segmentation: Facade Segment, Side Wall at Level 2: chessboard: 25
      v. classification: Smooth Flat at Level 2: Flat Architectural Component
      vi. classification: outlier_bay, Smooth Flat, Smooth Side, Smooth Vertical, unclassified at Level 2: Bay, Projection
      vii. image object fusion: loop: Bay at Level 2: <- Bay, Outlier, Smooth Flat, Smooth Side, Smooth Vertical, Step, unclassified: Bay + all Min. pixel value Range <= 0
           1. merge region: Facade Segment at Level 2: merge region
           2. assign class: Bay with Height < 0.2 and Depth < 1 at Level 2: unclassified
           3. classification: Facade Segment at Level 2: Facade Segment *
   b. Side-Wall-Relative (TBD)

11. export vector layers: at Level 2: export object shapes to ObjectShapes

12. Cleanup
   a. chessboard segmentation: Facade Segment, Side Wall at Pixel Level: chessboard: 50
   b. chessboard segmentation: unclassified at Pixel Level: chessboard: 1
   c. Spectral Segmentation Front and Side
      i. image object fusion: loop: Smooth Vertical, unclassified at Pixel Level: <- unclassified all side-look range <= 0.1
      ii. image object fusion: loop: Smooth Side, Smooth Vertical, unclassified at Pixel Level: <- unclassified all ATD Dist. From -10deg Line <= 0.1
      iii. image object fusion: loop: Facade Segment, unclassified at Pixel Level: <- unclassified all side-look range <= 0.1
   d. image object fusion: loop: Side Wall at Pixel Level: <- unclassified all ATD Dist. From -10deg Line <= 0.1
   e. assign class: unclassified with XY Vector < 0.2 at Pixel Level: Outlier
   f. assign class: unclassified with Height < 0.2 at Pixel Level: Outlier
   g. assign class: unclassified with Width <= 2 Pxl at Pixel Level: Outlier
   h. spectral difference segmentation: Smooth Side, Smooth Vertical, Step, unclassified at Pixel Level: spectral difference 1 creating 'Upper Level'
   i. Copy Up Classification
j. assign class: with Existence of sub objects Automobile Body (1) = 1 at Upper Level: Automobile Body
   i. assign class: with Existence of sub objects Curb (1) = 1 at Upper Level: Curb
k. assign class: with Existence of sub objects Side Wall (0) = 1 at Upper Level: Side Wall
l. assign class: with Existence of sub objects Facade Segment (1) = 1 at Upper Level: Facade Segment
m. assign class: with Existence of sub objects Surface (1) = 1 at Upper Level: Surface
   i. assign class: with Existence of sub objects Roof (1) = 1 at Upper Level: Roof
n. find enclosed by image object: Roof at Upper Level: enclosed by domain: Roof
o. assign class: unclassified with Existence of super objects Roof (1) = 1 at Pixel Level: Roof
p. merge region: Roof at Pixel Level: merge region
q. [image object fusion: Sign, Smooth Flat, Smooth Side, Smooth Sloped, Smooth Vertical, Step, unclassified at Level 2: <- Sign, Smooth Flat, Smooth Side, Smooth Sloped, Smooth Vertical, Step, unclassified: unclassified + all side-look range <= 1]
r. [assign class: Sign, Smooth Flat, Smooth Side, Smooth Sloped, Smooth Vertical, Step at Level 2: unclassified]
s. [copy map: copy map to 'backup2']
t. [copy map: copy map to 'main']
u. chessboard segmentation: unclassified at Pixel Level: chess board: 1
v. merge region: Side Wall at Level 2: merge region
w. chessboard segmentation: Outlier, outlierprojection at Pixel Level: chessboard: 1
x. assign class: outlier_bay, outlier_projection at Pixel Level: Outlier
y. merge region: Surface at Level 2: merge region
z. copy image object level: at Pixel Level: copy creating 'lower pixel' below
aa. chessboard segmentation: at lower pixel: chess board: 1
bb. assign class: Flat Architectural Component at Pixel Level: Smooth Flat

13. Export Objects
   a. export object statistics: Facade Segment with Height >= 3 at Level 2: export object statistics
   b. export object statistics: Side Wall at Level 2: export object statistics
   c. export object statistics: Surface at Level 2: export object statistics
   d. export object statistics: Car, VanBus, VanTruckSuv at Level 2: export object statistics
   e. export object statistics: Car, VanBus, VanTruckSuv at Level 2: export object statistics
   f. export object statistics: Sign at Level 2: export object statistics
   g. export object statistics: with Existence of super objects Bay (2) = 1 at lower pixel: export object statistics
Appendix E

Sign Recognizer: Procedural Workflow Diagram