University of Alberta

As-built Invert Modeling and Visualization for Tunnels by Using TBM Positioning Data

by

Xiaodong Wu

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

in

Construction Engineering and Management

Department of Civil and Environmental Engineering

Spring 2014
Edmonton, Alberta

Abstract

With the trend of global urbanization, underground space is being more and more exploited. To fulfill the need of sustainable development of underground space, it is necessary to well document the as-built positions of the facilities we have built. In the current practice, the as-built models are generated either by using remote sensing technologies like 3D Laser Scanning and Ground Penetrating Radar (GPR), or directly measuring the as-built positions of the facilities. However, there are several limitations of these approaches, such as considerable time and efforts required for modeling, high cost and low accuracy. This research proposes a new approach for the as-built invert modeling and visualization of bored tunnels, by using real-time Tunnel Boring Machine (TBM) tracking and positioning data. The main contribution of this research is the formalization and implementation of a new approach for the as-built invert modeling of tunnels.

Acknowledgement

First of all, I would like to gratefully thank my supervisor Dr. Ming Lu. His vision, knowledge, and dedication to high standards of work have influenced me throughout my MSc study at University of Alberta. His support, guidance and encouragement are also deeply appreciated.

Next, I would like to acknowledge my committee members, Dr. Simaan Abourizk and Dr. Lijun Deng, for their time reviewing my work and also for the advice and suggestions on improving the quality of this thesis.

I would also like to thank Dr. Johnson Shen and Mr. Sheng Mao, for the time we spent together in doing field trials and for their efforts in making the VLTB TBM Guidance System a reality.

My thanks also go to all the people who helped me through my graduate program. What I learned from my 2 years' study, were not only knowledge and skills, but also tenacity and passion on work, and life.

Special thanks to engineers, surveyors and technicians from the Design and Construction Section of Drainage Services at City of Edmonton. Without their support and assistance, the field trials would not be possible. Their support and assistance also helped me to obtain more knowledge in tunnel construction.

Finally, I would like to thank my mom and dad, for their continuous support, encouragement, and love.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1: INTRODUCTION

1.1 Overview

With the trend of global urbanization, problems such as traffic congestion, air pollution, lack of green space, and insufficient water supplies are becoming more and more common all around the world (Durmisevic 1999, Maire and Blunier 2006). To solve these problems, to sustain our society, and to improve the quality of life, we have built a great amount of underground facilities, such as tunnels, utility lines, underground walkways, et al. (ITA 2010).

As we further develop the underground space, we also make it increasingly congested. Figure 1-1 is a conceptual drawing of the Lidabashi Station in Tokyo, Japan, in which we can see several metro lines converged in a very small area.

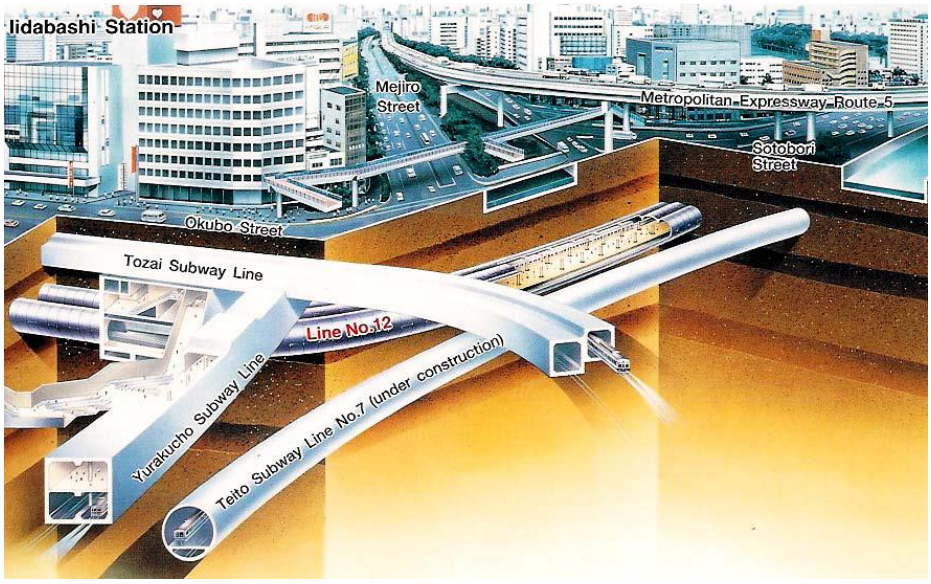


Figure 1-1: Tokyo Subway – Lidabashi Station, Tokyo, Japan (Courtesy: ITA)

What does it mean by a congested underground space? Compared with the above-ground world in which we live, work, and travel every day, the underground world is invisible, which results in a risk of collisions between existing underground facilities and the excavating machinery [e.g. excavators or Tunnel Boring Machines (TBM)]. According to Pipeline and Hazardous Materials Safety Administration of U.S. Department of Transportation (2013), over the last 20 years, there were in total 1948 pipeline incidents caused by excavation-related damage, resulting in 146 fatalities, 505 injuries, and over $472 million in property damage.

To minimize such a risk, it is necessary that the contractors know exactly where the existing utilities are buried. In the current practice in Alberta, a non-profit corporation called Alberta One-Call provides services of locating buried utilities to contractors. The "locate" process is illustrated in Figure 1-2.

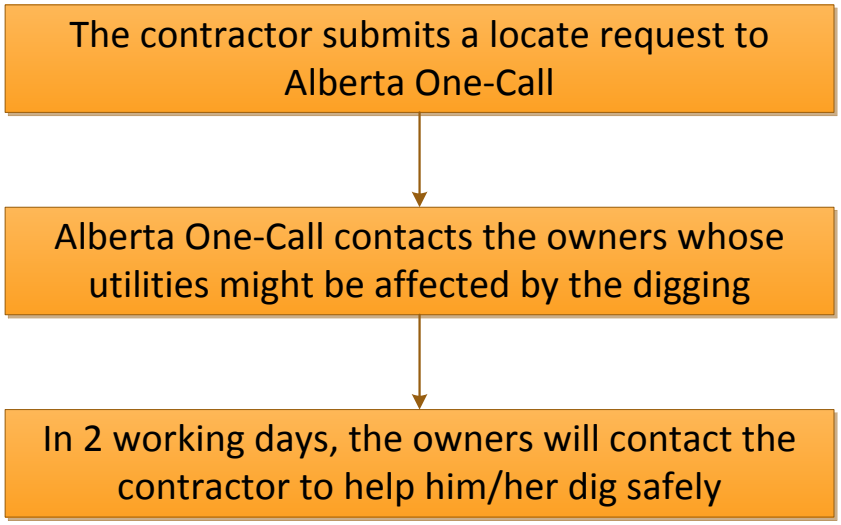


Figure 1-2: The locate process of Alberta One-Call [adapted from (Alberta One-Call 2013)]

The locate process of Alberta One-Call, with proper implementation, is effective for reducing excavation-related damages to buried utilities. However, it is still not efficient enough, as the process requires the involvement of the owners, the available as-built records can be incomplete or inaccurate, and it often results in a two working days' delay for the contractor.

With recent developments in Information Technology, especially Geographic Information Systems (GIS), several utility positioning data management systems have been developed, such as VISTA (Beck et al. 2007, Beck et al. 2009), and 3D GIS in the Cloud (Sivan Design 2013). These systems are capable of integrating utility positioning information from multiple sources and visualizing the utilities in a user-friendly way. By adopting these up-to-date management systems, the efficiency of locating buried utilities can be greatly improved.

Nonetheless, there is still one problem remaining to be solved. That is, how can we get as-built positioning data of the utilities, accurately and cost-effectively? Without accurate as-built positioning data, even the most advanced positioning data management system would become unreliable.

In addition, for some underground facilities (e.g. tunnels), the construction quality assurance standard is extremely high. According to Megaw and Bartlett (1981), the tolerance for the as-built alignment of a metro tunnel can be as tight as ±40 mm. For a drainage tunnel, the tolerance is normally limited to ±50 mm in both horizontal and vertical directions; the maximum deviation over the total length of a tunnel (usually a few kilometers) must be controlled within less than 150 mm (The City of Edmonton 2012). As we cannot intuitively see the TBM and the as-built tunnel, keeping the tunnel alignment in tight margins along such a long distance is like driving a car in complete

darkness, which is really a challenging task. To ensure construction quality, it is desirable to monitor the position of the as-built tunnel in real time with mm level accuracy as tunneling operations unfold.

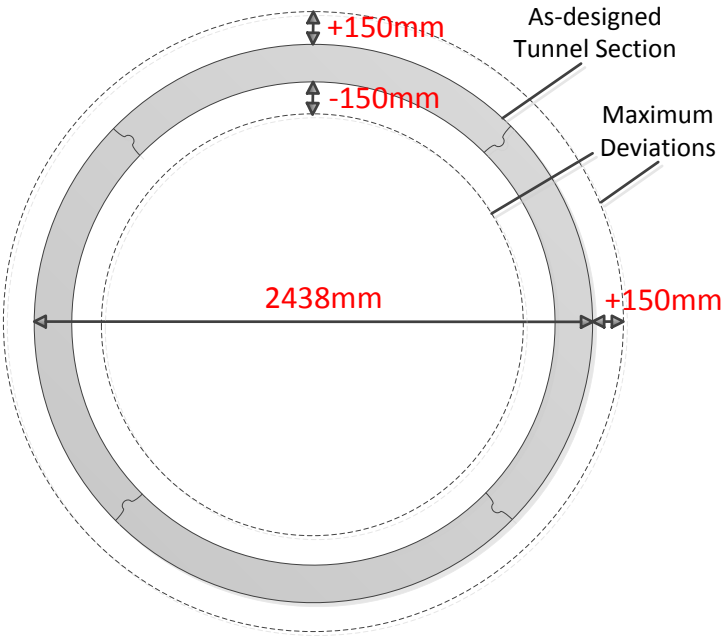


Figure 1-3: Dimension of a tunnel cross section and maximum deviations

In short, as we build more and more underground facilities, knowing what and where we have built underground is necessary for reducing excavation-related damages to existing underground facilities, while it is also important to monitor the position of the underground facilities being constructed for quality assurance.

1.2 Research Objectives and Scope

Just like the above ground world that we are familiar with, there are also various kinds of underground facilities, like pipelines, utility tunnels, transit tunnels, metro stations, underground car parks, and underground shopping malls. For different facilities, the structural materials might be different (e.g. a pipeline can be produced from plastic, metal,

or concrete); the construction methods might be different (e.g. a tunnel can be built by TBM method, drill and blast method, or hand digging method); the depths might be different [e.g. from within one meter to more than a hundred meters (Tng 2012)]; the dimensions might also be different [e.g. the diameter of a pipeline can be as small as a few decimeters, while the diameter of a transit tunnel can be as big as 18.65m (Grübl 2012)]. The differences on structural materials, construction methods, depths and dimensions make it hard to propose a single as-built modeling method that is applicable to all underground facilities.

In this thesis, the objective is to develop an as-built invert modeling and visualization system for tunnels built by TBM method. The positions of the inverts are often the most important as-built measurements for quality assurance of tunnelling. In addition, for tunneling in urban areas, TBMs are often preferred to other construction methods, due to their high advance rate, precise as-built profile and low impacts to buildings and traffic on the ground (Maidl et al. 2012). Thus, the proposed approach is applicable for a large proportion of underground utilities.

To reach the research objective, the Virtual Laser Target Board (VLTB) TBM Guidance System (Shen et al. 2012) is used for guiding the TBM as well as positioning data acquisition. A traditional laser guidance system is only able to obtain the line and grade deviations of a TBM. The VLTB system is capable to determine the three orientation angles (Yaw, Pitch, Roll), by using a robotic total station to automatically track and survey three prisms installed in a visible corner at the rear end of the TBM. Then, a point-to-angle algorithm is evoked to analytically fix the three orientation angles based on the positioning data from the total station. A tablet is used to control the automatic operations of the total station, execute the point-to-angle algorithm, and display the deviations of the TBM to the TBM operator on the fly. It is noted that a ZigBee wireless communication

network is applied for data communication between different components of the VLTB system.

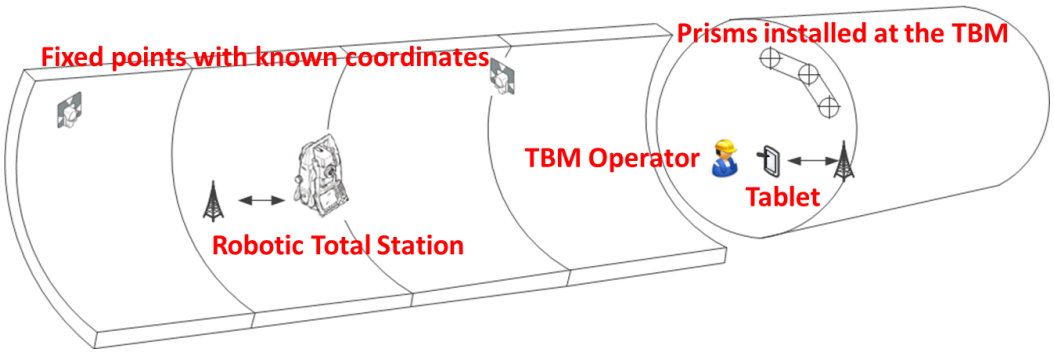


Figure 1-4: System architecture of VLTB (Courtesy: Sheng Mao)

In the research, a database is built for storing both positioning data and project-related data such as soil layers. A computer program is further developed on Microsoft XNA Framework for as-built invert modeling and visualization. With the VLTB system being implemented, the TBM is turned into a "sensor" to map as-built invert information in real time, without incurring extra labor cost or surveying equipment. The main contribution of this research is the formalization and implementation of a new approach for the as-built invert modeling and visualization of tunnels based on a TBM tracking and positioning system such as VLTB.

1.3 Thesis Organization

The remaining chapters of the thesis are organized as follows.

Chapter 2 begins with an introduction to TBM Tunnelling operations. The following section reviews related as-built modeling technologies that represent state of the art, such as laser scanning and Ground Penetrating Radar (GPR). The advantages and limitations of these technologies are also discussed.

In Chapter 3, first, the proposed algorithm is illustrated by several conceptual drawings. Then a mathematical analysis is conducted to define the modeling algorithm thoroughly and accurately. In the end of this chapter, a numerical example is given.

Chapter 4 focuses on technical details on how to implement the modeling algorithm. The overall system architecture is introduced, the structure of the database is presented, and some technical issues on how to visualize soil layers and the as-built tunnel are also discussed.

To validate the feasibility of the proposed as-built modeling method, in Chapter 5, a case study is carried out on a real-world drainage tunnel project in Edmonton, Alberta and results are presented. An error analysis is also conducted to validate its accuracy.

In the last Chapter 6, the limitations of the proposed as-built modeling approach and future improvements are discussed.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

In this chapter, firstly, TBM tunneling operations are introduced. Related research and practice on the as-built modeling and mapping of underground structures are then presented. Generally speaking, these efforts can be categorized into two groups, namely, remote sensing technologies like 3D Laser Scanning and GPR, or directly measuring the translations and rotations of concrete lining segments. The advantages and limitations of each approach are also discussed.

2.2 TBM Tunneling Operations

The first TBM, called mountain slicer, was built in 1846 by Henri-Joseph Maus, for a railway tunnel between France and Italy through the Alps (Hapgood 2004). After more than 100 years' technological advancement, various types of TBMs have been developed for tunneling in different ground conditions, as summarized in Figure 2-1.

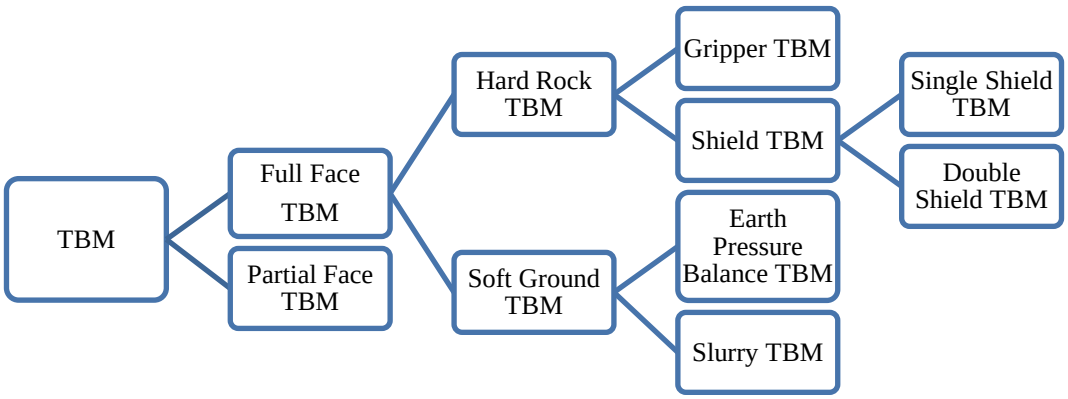


Figure 2-1: Types of TBM

A section view drawing of an Earth Pressure Balance (EPB) TBM, with its main components annotated, is illustrated in Figure 2-2. The cutter head (1) is designed to rotate around the central axis of the machine and excavate soils into the excavation chamber (2). The screw conveyor (5) moves soil out of the chamber. The thrust cylinders (4) are the key component to push the TBM forward. Every time the TBM advances for a distance equals to the width of a concrete lining segment, the thrust cylinders stop pushing the TBM against the pre-installed ring of concrete lining segments and are reset to their initial positions in the TBM. Then, the segment erector (6) installs new concrete lining segments (7). After a new ring of concrete lining segments are installed, a new cycle of TBM's advancing and concrete lining segments installation begins and the thrust cylinders continues to push the TBM forward.

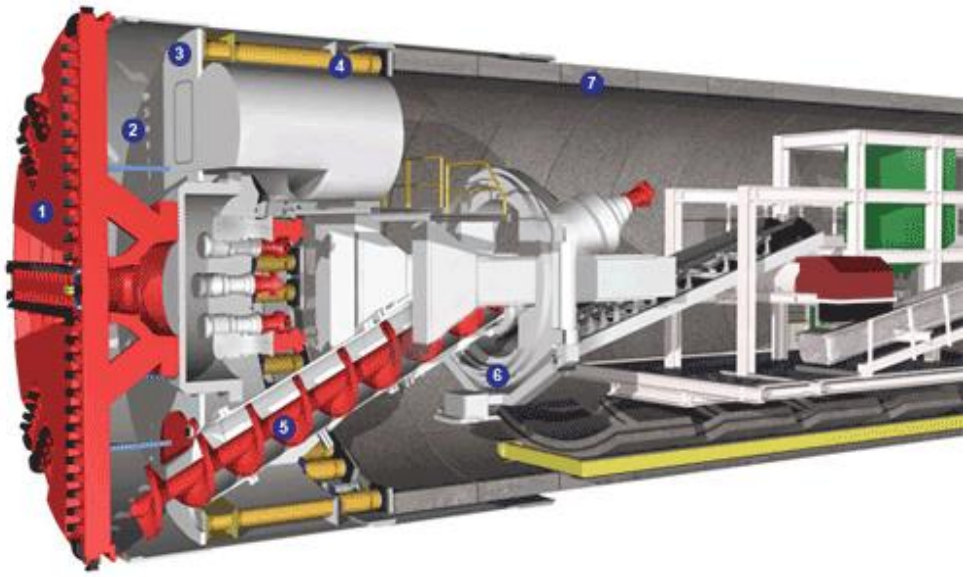


Figure 2-2: Illustration of EPB TBM (Courtesy: Herrenknecht)

Note: (1) Cutter head; (2) excavation chamber; (3) bulkhead; (4) thrust cylinders; (5) screw conveyor; (6) segment erector; and (7) concrete lining segment

Figure 2-3: Inner view of a tunnel under construction by a TBM

Usually, an EPB TBM is used in cohesive soils (such as clay) below the table of ground water. For tunneling in less cohesive soil and hard rock, Slurry TBM and hard rock TBM are used, respectively. The working mechanisms of these TBMs are similar to that of an EPB TBM except that the systems for removing soil or rock are different.

2.3 Technologies Applied for As-Built Modeling of Tunnels

2.3.1 3D Laser Scanning

3D Laser Scanning is a technology that profiles the shape of objects within line-of-sight by emitting laser beams and measuring their return. It is a highly automated and advanced technology that is able to obtain a great amount of spatial information in the form of point clouds in a short period of time (Klein, Li, and Becerik-Gerber 2012).

A 3D laser scanner can be used for various applications in the domain of civil engineering, such as landslide monitoring, highway earthwork assessment, bridge

inspection, et al. (Schaefer, Burckhard, and Boomer 2005; Duffell and Rudrum 2005; Tang and Akinci 2012) It can also be used for many applications in tunnelling, such as as-built modeling, tunnel lining evaluation, rock mass discontinuity evaluation, and deformation detection (Fekete, Diederichs, and Lato 2010; van Gosliga, Lindenbergh, and Pfeifer 2006).

Figure 2-4: A Leica HDS-3000 Laser Scanner (Courtesy: David Monniaux)

According to Ghassemi, Zoldy, and Javady (2010) and Klein, Li, and Becerik-Gerber (2012), the advantages and limitations of applying 3D laser scanning for the as-built modeling of tunnels are listed in Table 2-1.

Table 2-1: Advantages and limitations of laser scanning

Advantages	Limitations
1. High accuracy (millimeter level) 2. High resolution 3. Automated data collection 4. Excellent way for presentation	1. High equipment cost 2. Non-portable 3. High requirement on skills 4. Strict requirement on line of sight

For a typical tunnel project, 3D laser scanning is only possible after project completion, as there are many temporary facilities in the congested tunnel space in the construction stage, blocking the line of sight of the laser scanner, as shown in Figure 2-5.

Figure 2-5: Typical environments inside a tunnel under construction

2.3.2 GPR

GPR is a geophysical method that is capable of locating underground structures by transmitting high-frequency radio waves into the ground and detecting reflected signals from underground structures. Since the 1970s, it has been applied in various areas, such as environmental and agricultural monitoring, glaciological monitoring, landmine detection, archaeological investigations and civil engineering (Metje et al. 2007).

A GPR is able to locate a wide variety of buried pipes, from plastic pipes to metallic pipes. However, this technology is limited in several aspects. First, the performance of a GPR is unstable. It largely depends on the electrical conductivity of soil at the site. If the conductivity is high, the radar signal attenuates very fast in the ground and the maximum

penetration range would be significantly reduced. Second, the modeling accuracy is low. Under normal conditions, a GPR can achieve foot-level accuracy, which is not enough for accurate as-built modeling. Third, interpretation of a radar-gram is not simple and requires special training, as it is full of noises and uncertainties. (Ghassemi, Zoldy, and Javady 2010)

To improve the performance of GPR, various attempts have been made. Metje et al. (2007) developed a multi-sensor locating tool combining GPR with quasi-static fields and acoustics; Young et al. (2009) combined GPR with differential GPS positioning, robotic total-station, and software systems to create 3D model of underground structure and utility lines; Zou et al. (2012) applied a new design of an ultra-wideband (UWB) radar with a higher sensing accuracy than traditional GPR in detecting and imaging of buried objects underground in support of open-cut or tunnel applications.

However, as denoted by Costello (2007), no single technology, or combination of technologies, is able to locate all existing underground utilities with 100 percent accuracy. Zou et al. (2012) also identified that one disadvantage of UWB radar is the high signal loss in the ground (even higher than traditional GPR), which results in a limited penetration range.

2.3.3 Direct Measurement

In addition to remote sensing technologies mentioned above, a more straightforward approach is to directly measure the translations and rotations of concrete lining segments, as described in Li and Zhu (2009).

The direct measurement approach worked well in Shanghai Yangtze River Tunnel, a large twin-tube bored tunnel with an outer diameter of 15.0m (Li, Zhu, and Zhen 2009).

However, for a typical tunnel project, especially a small diameter one, it is not easy to apply their approach as translation and rotation information of concrete lining segments is not readily available. Besides, the manual process of measuring the positions of lining segments is time-consuming and requires the expertise of specialist surveyors.

CHAPTER 3: PROPOSED AS-BUILT INVERT MODELING APPROACH FOR TUNNELS

To improve the current practice, a new approach is proposed based on real-time TBM positioning data. The proposed approach is first illustrated with several conceptual drawings, and then the underlying mathematical foundations are presented. In the end, a numerical example is also illustrated. Although the mathematical analysis in this Chapter is based on input data sourced from VLTB, it is generic and can be readily adapted to other TBM guidance systems.

3.1 Illustration of the Proposed Methodology

In a tunneling site, a tunnel is built by installing concrete segments behind the TBM ring by ring, just like a masonry wall is built by laying bricks layer by layer. It is reasonably assumed that the position of the latest installed tunnel section is determined by the position (distance, line deviation, and grade deviation) and orientation angles (yaw and pitch angles) of the TBM, as shown in Figure 3-1. It is also worth noting that theoretically, the roll angle has no effect on the profile of the latest installed tunnel section, as shown in Figure 3-2.

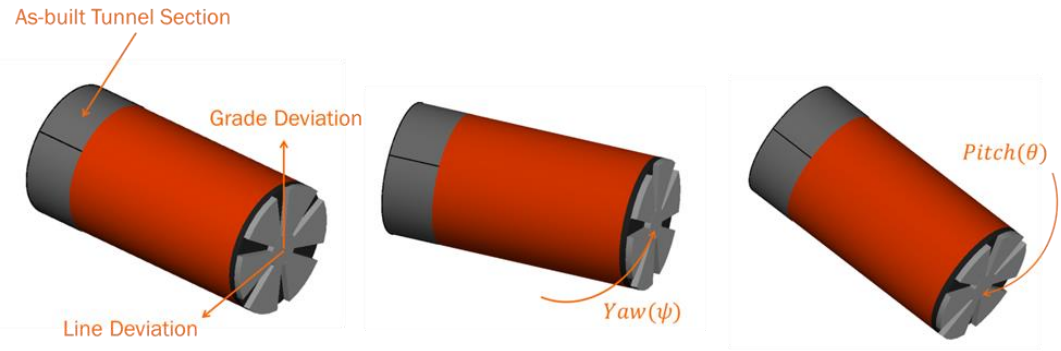


Figure 3-1: The latest installed tunnel section and the position and orientation angles of the TBM

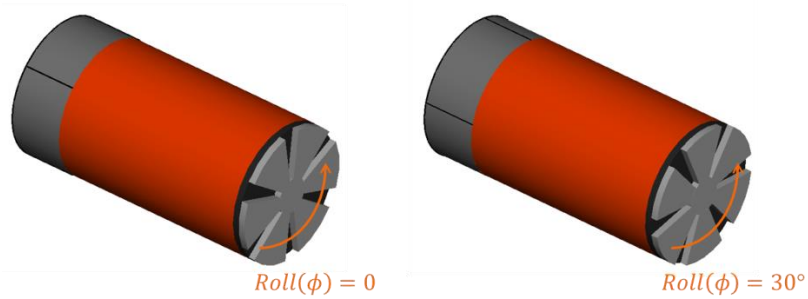


Figure 3-2: Effects of roll angle on the latest installed tunnel section

As the TBM moves forward, new tunnel sections are installed one by one along the advancing direction of the machine. Correspondingly, the as-built tunnel model is also updated based on the latest TBM positioning data available, as illustrated in Figure 3-3. The positions of the as-built inverts are simply the lowest points along the as-built tunnel.

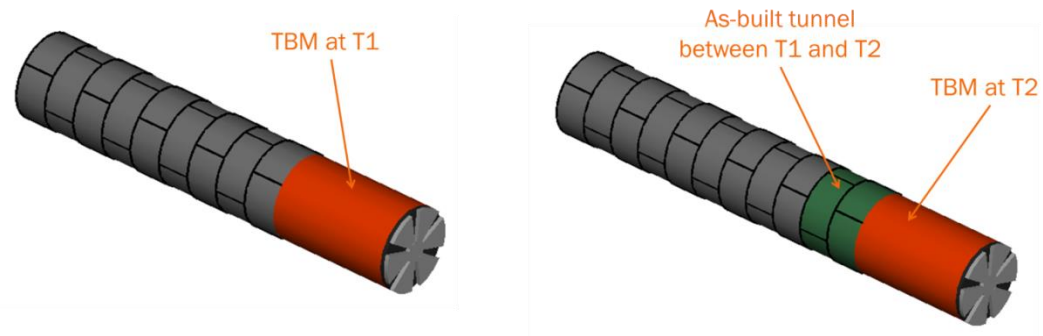


Figure 3-3: Updating the as-built tunnel model

3.2 Mathematical Foundations

In this section, the underlying mathematical foundations are presented in details.

3.2.1 Coordinate Systems

Three Cartesian coordinate systems are defined, namely, the local geodetic frame (F_n, n-frame), the TBM's body frame (F_b, b-frame), and the tunnel frame (F_t, t-frame), as shown in Figure 3-4. The n-frame is usually the standard frame in construction surveying. The origin of the frame is fixed in a particular location and the three axes are along the east, north, and geodetic zenith. The b-frame is fixed on the center of the TBM, with (1) Y-axis along the TBM's advancing direction, (2) X-axis perpendicular to the Y-axis and parallel to the horizontal plane, and (3) Z-axis aligned along the cross product of X-axis and Y-axis (Shen, Lu and Chen 2011). In addition, to make visualization easier, the t-frame is also defined, with (1) the origin on the projection of the starting point of the tunnel on the geoid (a geoid is a terminology in surveying, which is the shape of the sea level under the influence of earth's gravity alone) (2) Y-axis being parallel to the projection of the advancing direction of the as-designed tunnel at the starting point on the horizontal plane (3) Z-axis aligned along the geodetic zenith, and (4) X-axis along the cross product of Y-axis and Z-axis.

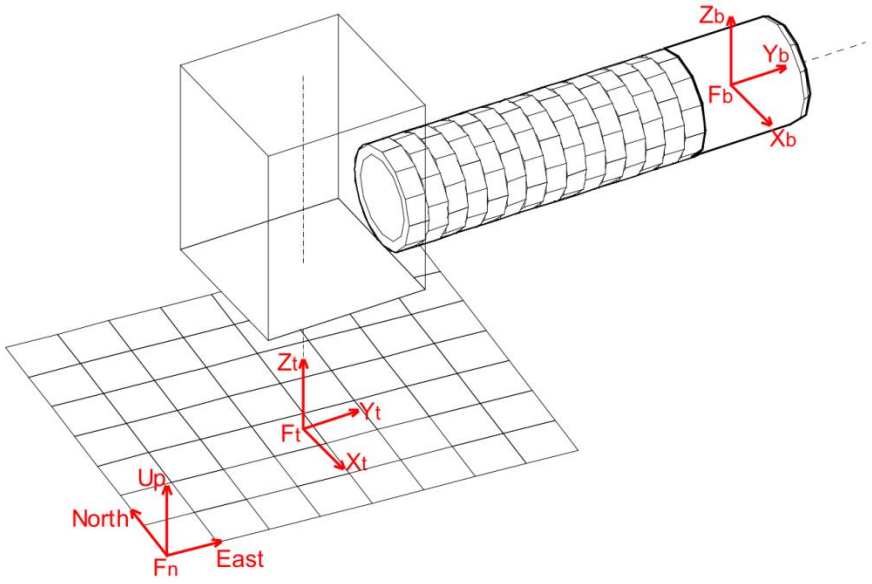


Figure 3-4: The local geodetic frame, TBM's body frame and tunnel frame

The reason for defining multiple frames, instead of one single frame, is that different coordinate systems have different application purposes. As mentioned above, the n-frame is usually the standard frame in construction surveying. To minimize frame transformations between different projects, it is desirable to keep our as-built invert records in the n-frame. On the other hand, to visualize the construction of a certain tunnel project, the t-frame is usually the most convenient one to apply, as all the spatial information in the t-frame is referenced to the starting point of the tunnel. In other words, all the spatial information in a t-frame is localized to this tunnel project.

3.2.2 The Transformation between the n-frame and the b-frame

According to Shen, Lu, and Chen (2011), given the TBM's three rotation angles, Yaw (ψ), Pitch (θ) and Roll (ϕ), the transformation matrix from the b-frame to the n-frame can be expressed as Equation (3-1):

$$C_b^n = \begin{pmatrix} \cos\phi\cos\psi - \sin\phi\sin\theta\sin\psi & \cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & -\sin\phi\cos\theta \\ -\cos\theta\sin\psi & \cos\theta\cos\psi & \sin\theta \\ \sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \sin\phi\sin\psi - \cos\phi\sin\theta\cos\psi & \cos\phi\cos\theta \end{pmatrix} \quad (3\text{-}1)$$

With the transformation matrix C_b^n, coordinates in the b-frame can be transformed to coordinates in the n-frame by Equation (3-2):

$$\hat{p}_n = C_b^n \hat{p}_b \quad (3\text{-}2\ a)$$

$$\hat{v}_n = C_b^n \hat{v}_b \quad (3\text{-}3\ b)$$

Where $\hat{p}_b$ and $\hat{v}_b$ denote the three dimensional coordinates of a point p and vector v in the b-frame, $\hat{p}_n$ and $\hat{v}_n$ denote the counterparts in the n-frame.

Correspondingly, coordinates in the n-frame can be transformed back to the b-frame by Equation (3-3):

$$\hat{p}_b = \left(C_b^n\right)^{-1} \hat{p}_n \tag{3-4 a}$$

$$\hat{v}_b = \left(C_b^n\right)^{-1} \hat{v}_n \tag{3-5 b}$$

However, the TBM's three rotation angles are not readily available, without using inclinometers or other angle-measurement instruments. The following deterministic triaxis attitude determination (TRIAD) algorithm (Shen, Lu, and Chen 2011) is applied in VLTB to calculate the three rotation angles and the transformation matrix C_b^n,.

Firstly, as mentioned in Chapter 1, three non-collinear prisms P_1, P_2 and P_3, are installed in a visible corner at the rear end of a TBM, as shown in Figure 3-5. The three dimensional coordinates of these points in the b-frame, $\hat{p}_{1b}$, $\hat{p}_{2b}$, and $\hat{p}_{3b}$, can be measured directly from the TBM, while the corresponding coordinates in the n-frame, $\hat{p}_{1n}$, $\hat{p}_{2n}$, and $\hat{p}_{3n}$, can be retrieved by a total station that is set up in the n-frame.

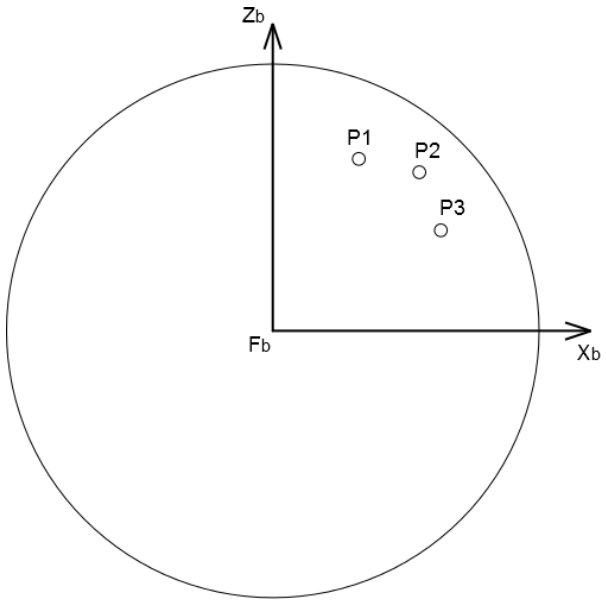


Figure 3-5: Prisms installed at the rear end of a TBM

Next, the two vectors $\hat{v}_1$ and $\hat{v}_2$ are defined in the b-frame as Equation (3-4):

$$\begin{cases} \hat{v}_1 = \hat{p}_{2b} - \hat{p}_{1b} \\ \hat{v}_2 = \hat{p}_{3b} - \hat{p}_{1b} \end{cases} \tag{3-6}$$

Three orthogonal and unified vectors $\hat{r}_1$, $\hat{r}_2$, and $\hat{r}_3$ are further determined as Equation (3-5):

$$\begin{cases} \hat{r}_1 = \hat{v}_1 / \left\| \hat{v}_1 \right\| \\ \hat{r}_2 = \hat{v}_1 \times \hat{v}_2 / \left\| \hat{v}_1 \times \hat{v}_2 \right\| \\ \hat{r}_3 = \hat{r}_1 \times \hat{r}_2 \end{cases} \tag{3-7}$$

Similarly, given the corresponding coordinates of P_1, P_2 and P_3 in the n-frame as $\hat{p}_{1n}$, $\hat{p}_{2n}$, and $\hat{p}_{3n}$, the two vectors $\hat{w}_1$ and $\hat{w}_2$ and the three orthogonal and unified vectors $\hat{s}_1$, $\hat{s}_2$, and $\hat{s}_3$ are also defined, according to the same rules in Equation (3-4) and Equation (3-5).

With $\hat{r}_1$, $\hat{r}_2$, $\hat{r}_3$ and $\hat{s}_1$, $\hat{s}_2$, $\hat{s}_3$ determined, two orthogonal 3×3 matrices M_b and M_n are constructed as given in Equation (3-6) and Equation (3-7):

$$M_b = \begin{pmatrix} \hat{r}_1 & \hat{r}_2 & \hat{r}_3 \end{pmatrix} \tag{3-8}$$

$$M_n = \begin{pmatrix} \hat{s}_1 & \hat{s}_2 & \hat{s}_3 \end{pmatrix} \tag{3-9}$$

It is worth mentioning that Equation (3-2) also applies to M_b and M_n. Substituting M_b and M_n to Equation (3-2), we have:

$$M_n = C_b^n M_b \tag{3-10}$$

With Equation (3-8), C_b^n can be solved by Equation (3-9):

$$C_b^n = M_n \left(M_b\right)^{-1} \tag{3-11}$$

Because M_b is proper orthogonal

$$\left(M_b\right)^{-1} = M_b^T \tag{3-12}$$

Substituting Equation (3-10) to Equation (3-9), we have

$$C_b^n = M_n M_b^T \tag{3-13}$$

With the transformation matrix C_b^n determined by Equation (3-11), finally, the three rotation angles, Yaw (ψ), Pitch (ϕ) and Roll (ϑ) can be calculated by Equation (3-12):

$$\begin{cases} \psi = -\arctan(C_b^n(2,1)/C_b^n(2,2)) \\ \theta = \arcsin(C_b^n(2,3)) \\ \varphi = -\arctan(C_b^n(1,3)/C_b^n(3,3)) \end{cases} \tag{3-14}$$

Where $C_b^n(i, j)$ represents the element in row i and column j of matrix C_b^n .

However, the matrix C_b^n is a linear transformation. It can be used for rotations between different frames, but not for translations or a combination of rotations and translations. To make the transformation matrix more general, homogeneous coordinates and affine transformations, which are common practice in 3D Computer Graphics (Guha, 2011), are introduced. The new affine transformation matrix is given as Equation (3-13):

$$F_b^n = \begin{pmatrix} C_b^n & T_b \\ 0 & 1 \end{pmatrix} \tag{3-15}$$

Where T_b represents a column vector $(E_b \quad N_b \quad U_b)^T$ (E_b , N_b and U_b denote the Easting, Northing and Up coordinates of the center of the TBM, respectively). The original 3×3 matrix C_b^n is extended to a 4×4 matrix, by adding a row vector zero to the left bottom of C_b^n , a column vector T_b to the top right of C_b^n , and an element 1 to the right bottom. With Equation (3-13), Equation (3-2) can be rewritten as:

$$\begin{pmatrix} \hat{p}_{nE} \\ \hat{p}_{nN} \\ \hat{p}_{nU} \\ 1 \end{pmatrix} = F_b^n \begin{pmatrix} \hat{p}_{bx} \\ \hat{p}_{by} \\ \hat{p}_{bz} \\ 1 \end{pmatrix} \tag{3-16 a}$$

$$\begin{pmatrix} \hat{v}_{nE} \\ \hat{v}_{nN} \\ \hat{v}_{nU} \\ 0 \end{pmatrix} = F_b^n \begin{pmatrix} \hat{v}_{bx} \\ \hat{v}_{by} \\ \hat{v}_{bz} \\ 0 \end{pmatrix} \tag{3-17 b}$$

In Equation (3-14 a), $\hat{p}_n$ and $\hat{p}_b$ are extended from the original 3×1 column vectors to 4×1 column vectors, by adding an element 1 to the bottom of each vector. While in Equation (3-14 b), $\hat{v}_n$ and $\hat{v}_b$ are also extended to 4×1 column vectors by adding an element 0 to the bottom of each vector.

Nonetheless, there is still one problem remaining to be solved, that is, how can we obtain the coordinates of the center of the TBM in the n-frame? In the remaining part of this section, an extension of the TRIAD algorithm in the form of homogeneous coordinates and affine transformations is presented.

Given the two orthogonal 3×3 matrices M_b and M_n determined in Equation (3-6) and Equation (3-7), the following two matrices A_b and A_n are defined as Equation (3-15) and Equation (3-16):

$$A_b = \begin{pmatrix} M_b & \hat{p}_{1b} \\ 0 & 1 \end{pmatrix} \tag{3-18}$$

$$A_n = \begin{pmatrix} M_n & \hat{p}_{1n} \\ 0 & 1 \end{pmatrix} \tag{3-19}$$

Similar to the case of C_b^n in Equation (3-13), the original 3×3 matrices M_b and M_n are extended to 4×4 matrices A_b and A_n, by adding row vectors zeroes to the bottom of M_b and M_n, the column vectors $\hat{p}_{1b}$ and $\hat{p}_{1n}$ to the right of M_b and M_n respectively, and elements 1's to the right bottom. As defined in the earlier part of this section, $\hat{p}_{1b}$ and $\hat{p}_{1n}$ represent the three dimensional coordinates of Prism P1 in the b-frame and the n-frame, respectively.

Substituting A_b and A_n to Equation (3-14), we have:

$$A_n = F_b^n A_b \tag{3-20}$$

In which F_b^n can be solved by Equation (3-18):

$$F_b^n = A_n \left(A_b\right)^{-1} \tag{3-21}$$

With F_b^n determined by Equation (3-18), we can use Equation (3-14) to do transformations between the n-frame and the b-frame.

3.2.3 The transformation between the n-frame and the t-frame

The affine transformation matrix from the t-frame to the n-frame can be given as Equation (3-19):

$$F_t^n = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & E_t \\ \sin\alpha & \cos\alpha & 0 & N_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3-22}$$

Where α denotes a rotation of F_n around the Up axis. With such a rotation on F_n, the directions of E, N and U coincide with those of X_t, Y_t and Z_t, respectively. E_t and N_t represent the Easting and Northing coordinates of the starting point of the tunnel in the n-frame. As the origin of the t-frame lies on the geoid, the element $F_t^n(3,4)$ is zero. Similar to Equation (3-14), coordinates in the t-frame can be transformed to the n-frame by:

$$\begin{pmatrix} \hat{p}_{nE} \\ \hat{p}_{nN} \\ \hat{p}_{nU} \\ 1 \end{pmatrix} = F_t^n \begin{pmatrix} \hat{p}_{tx} \\ \hat{p}_{ty} \\ \hat{p}_{tz} \\ 1 \end{pmatrix} \tag{3-23 a}$$

$$\begin{pmatrix} \hat{v}_{nE} \\ \hat{v}_{nN} \\ \hat{v}_{nU} \\ 0 \end{pmatrix} = F_t^n \begin{pmatrix} \hat{v}_{tx} \\ \hat{v}_{ty} \\ \hat{v}_{tz} \\ 0 \end{pmatrix} \tag{3-24 b}$$

It is worth mentioning that by multiplying the invert of F_t^n on each side of Equation (3-20), we have:

$$\begin{pmatrix} \hat{p}_{tx} \\ \hat{p}_{ty} \\ \hat{p}_{tz} \\ 1 \end{pmatrix} = \left(F_t^n\right)^{-1} \begin{pmatrix} \hat{p}_{nE} \\ \hat{p}_{nN} \\ \hat{p}_{nU} \\ 1 \end{pmatrix} \tag{3-25 a}$$

$$\begin{pmatrix} \hat{v}_{tx} \\ \hat{v}_{ty} \\ \hat{v}_{tz} \\ 0 \end{pmatrix} = \left(F_t^n\right)^{-1} \begin{pmatrix} \hat{v}_{nE} \\ \hat{v}_{nN} \\ \hat{v}_{nU} \\ 0 \end{pmatrix} \tag{3-26 b}$$

Equation (3-21) transforms coordinates in the n-frame to the t-frame. Substituting Equation (3-14) to Equation (3-21), we get Equation (3-22):

$$\begin{pmatrix} \hat{p}_{tx} \\ \hat{p}_{ty} \\ \hat{p}_{tz} \\ 1 \end{pmatrix} = \left(F_t^n\right)^{-1} F_b^n \begin{pmatrix} \hat{p}_{bx} \\ \hat{p}_{by} \\ \hat{p}_{bz} \\ 1 \end{pmatrix} \tag{3-27 a}$$

$$\begin{pmatrix} \hat{v}_{tx} \\ \hat{v}_{ty} \\ \hat{v}_{tz} \\ 0 \end{pmatrix} = \left(F_t^n\right)^{-1} F_b^n \begin{pmatrix} \hat{v}_{bx} \\ \hat{v}_{by} \\ \hat{v}_{bz} \\ 0 \end{pmatrix} \tag{3-28 b}$$

Equation (3-22) transforms coordinates in the b-frame directly to the t-frame. Note that by using the n-frame as the frame of reference, the t-frame is fixed, while the b-frame changes over time according to the position and orientations of the TBM.

3.2.4 Interpolating the position and orientations of the TBM

All the positioning data retrieved from a TBM Guidance System (such as VLTB) are time-stamped (e.g. 2012-08-24 13:17:30) and discrete. However, when we make the as-

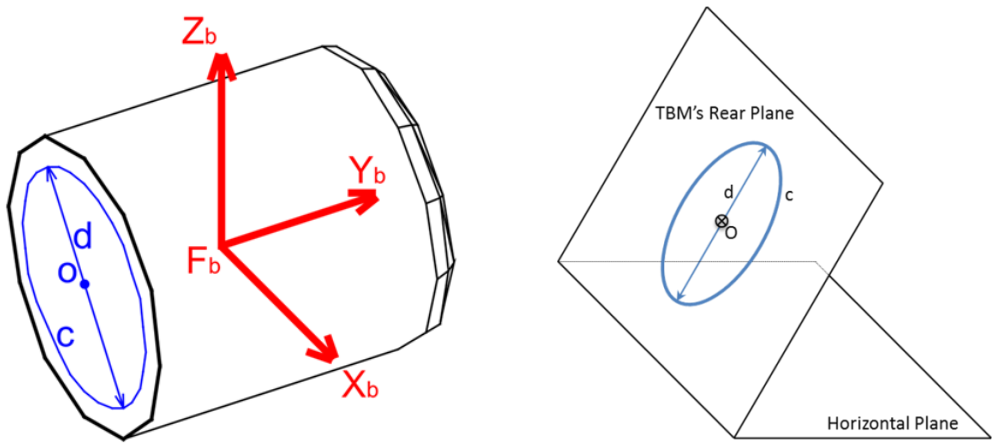
built invert model, we need positioning data between two points of time. Thus, it's necessary to perform interpolation.

Compared with other construction equipment such as excavators or trucks, a TBM moves slowly and is almost static. As mentioned in Shen et al. (2012), a typical advance rate of a TBM is 5m/shift (8 hours per shift), on average $1.74\times10^{-4} m/s$. The three orientation angles of the TBM are also stable. Based on field testing results in Shen, Lu, and Chen (2011), over an advance of 0.3m, the maximum fluctuation on the three orientation angles of a TBM is limited to about 1 degree.

As the velocity and angular velocity of a TBM are so low, there is no need to apply complex interpolations such as spherical linear quaternion interpolation that is commonly used in computer graphics for interpolating rotations of moving objects. Hence, linear interpolations of the three rotation angles and the position of the TBM are sufficient.

3.2.5 Modeling Algorithm

Given the position and orientation of the TBM at an arbitrary time t, we may draw a circle (*c*) on the rear plane of the TBM, as shown in Figure 3-6. The center of the circle (*o*) is on the center of the rear plane, and the diameter of the circle equals to the inner diameter of the as-designed tunnel (*d*). Suppose the concrete lining segments are installed accurately as designed. The as-built tunnel at t can be approximated by the circle *c* in Figure 3-6.



(a) (b)

Figure 3-6: The as-built tunnel at an arbitrary time t [(a) illustration in b-frame; (b) illustration with the horizontal plane]

Given the n-frame as the frame of reference, suppose that the normal vector and the center of the TBM's rear plane are known, namely, $\hat{n}_n$ and $\hat{o}_n$. In addition, the vector $\hat{u}_n$ is defined as the unit *up* vector in the n-frame [$(0 \quad 0 \quad 1)^T$], while the vector $\hat{h}_n$ is defined as the normalized cross product of $\hat{u}_n$ and $\hat{n}_n$, as Equation (3-23).

$$\hat{h}_n = \hat{u}_n \times \hat{n}_n / \left\| \hat{u}_n \times \hat{n}_n \right\| \tag{3-29}$$

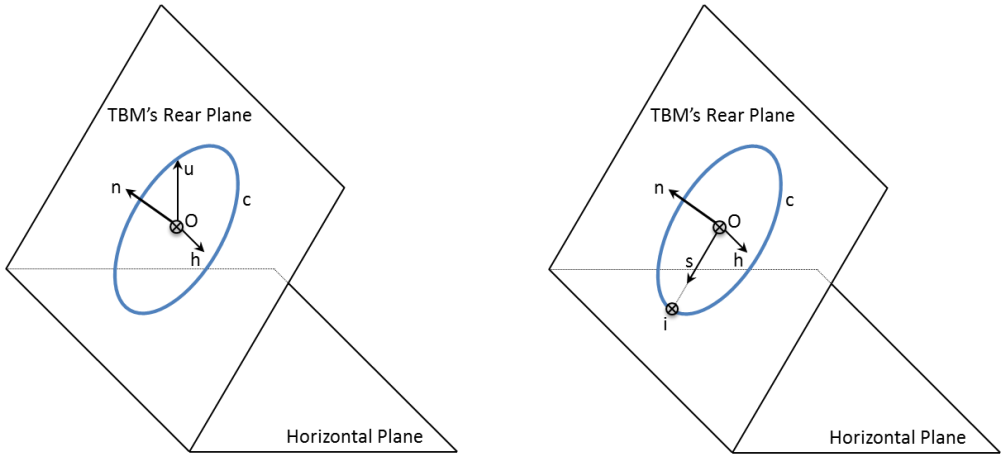
As its definition, $\hat{h}_n$ is perpendicular to both $\hat{u}_n$ and $\hat{n}_n$, which means that $\hat{h}_n$ is on the horizontal plane as well as the TBM's rear plane. By computing the normalized cross product of $\hat{h}_n$ and $\hat{n}_n$, we get another vector $\hat{s}_n$, as Equation (3-24).

$$\hat{s}_n = \hat{h}_n \times \hat{n}_n / \left\| \hat{h}_n \times \hat{n}_n \right\| \tag{3-30}$$

With $\hat{s}_n$ determined, the coordinates of the invert can also be calculated as Equation (3-25), where d is the inner diameter of the as-designed tunnel.

$$\hat{i}_n = \hat{o}_n + \frac{d}{2}\hat{s}_n \tag{3-31}$$

The whole process of determining the invert is shown in Figure 3-7.



(a) (b)

Figure 3-7: Determining the invert [(a) determining h from u and n; (b) determining s and i)]

However, in a real tunneling site, $\hat{n}_n$ and $\hat{o}_n$ are not readily available. Nonetheless, we can directly get coordinates of $\hat{n}$ and $\hat{o}$ in the b-frame, as Equation (3-26), where l is the length of the TBM.

$$\begin{cases} \hat{n}_b = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} \\ \hat{o}_b = \begin{pmatrix} 0 \\ -\frac{l}{2} \\ 0 \end{pmatrix} \end{cases} \tag{3-32}$$

Substituting $\hat{n}_b$ and $\hat{o}_b$ to Equation (3-14 b), we can get $\hat{n}_n$ and $\hat{o}_b$ as Equation (3-27)

$$\begin{cases} \begin{pmatrix} \hat{n}_n \\ 0 \end{pmatrix} = F_b^n \begin{pmatrix} \hat{n}_b \\ 0 \end{pmatrix} \\ \begin{pmatrix} \hat{o}_n \\ 1 \end{pmatrix} = F_b^n \begin{pmatrix} \hat{o}_b \\ 1 \end{pmatrix} \end{cases} \tag{3-33}$$

3.2.6 Numerical Example

The following numerical example illustrates how the as-built modeling approach works. The inputs are derived from a real world tunneling project with a straight as-designed alignment. The outputs are positions of the inverts at two separate time points t1 and t2. At both t1 and t2, the coordinates of the inverts in the n-frame are calculated.

Input data

Several spatial parameters are listed in Table 3-1.

Table 3-1: Spatial parameters

Parameters	Value
Tunnel inner diameter	2.340m
Tunnel total length	1012.835m
TBM outer diameter	2.526m
TBM length	4.652m
Tunnel starting point coordinates (in the n-frame)	(27739.751, 5935075.982, 643.080) (m)
Tunnel end point coordinates (in the n-frame)	(27744.650, 5934063.159, 644.090) (m)

Three prisms are installed on the rear end of the TBM, at P1, P2, and P3, as shown in Figure 3-5. The coordinates of these three points, in the b-frame, are listed in Table 3-2.

Table 3-2: Coordinates of prisms in the b-frame (unit: m)

Point	x	y	z
P1	0.408	–2.254	0.814
P2	0.695	–2.253	0.751
P3	0.797	–2.255	0.476

A set of positioning data at 13:17:15 (t1) and another set at 14:05:20 (t2) are simulated and shown in Table 3-3 and Table 3-4, respectively.

Table 3-3: Simulated position data at t1 (unit: m)

Point	Easting	Northing	Elevation
P1	27740.618	5934807.717	644.163
P2	27740.332	5934807.712	644.099
P3	27740.231	5934807.712	643.823

Table 3-4: Simulated position data at t2 (unit: m)

Point	Easting	Northing	Elevation
P1	27740.641	5934806.321	644.165
P2	27740.354	5934806.318	644.103
P3	27740.251	5934806.318	643.828

Transformation between the b-frame and the n-frame

Substituting the coordinates in **Table 3-2** into Equation (3-15), we have Equation (3-28).

$$A_b = \begin{pmatrix} 0.976739 & -0.005531 & 0.214362 & 0.408 \\ 0.003403 & 0.999941 & 0.010293 & -2.254 \\ -0.214406 & -0.009324 & 0.976700 & 0.814 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3-34}$$

Substituting the position data in **Table 3-3** and **Table 3-4** into Equation (3-16), we have the corresponding A_n at t1 and t2, respectively, as given in Equation (3-29) and Equation (3-30).

$$A_{n1} = \begin{pmatrix} -0.975723 & 0.019038 & -0.218180 & 27740.618 \\ -0.017058 & -0.999794 & -0.010954 & 5934807.717 \\ -0.218344 & -0.006967 & 0.975847 & 644.163 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3-35}$$

$$A_{n2} = \begin{pmatrix} -0.977401 & 0.011372 & -0.211087 & 27740.641 \\ -0.010217 & -0.999926 & -0.006564 & 5934806.32 \\ -0.211146 & -0.004259 & 0.977445 & 644.165 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3\text{-}36)$$

Substituting A_b, A_{n1} and A_{n2} to Equation (3-18), we have F_b^n at t1 and F_b^n at t2.

$$F_{b1}^n = \begin{pmatrix} -0.999901 & 0.013471 & -0.004073 & 27741.060 \\ -0.013480 & -0.999907 & 0.002280 & 5934805.46 \\ -0.004042 & 0.002335 & 0.999989 & 643.356 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3\text{-}37)$$

$$F_{b2}^n = \begin{pmatrix} -0.999977 & 0.005873 & 0.003286 & 27741.060 \\ -0.005856 & -0.999970 & 0.005102 & 5934804.06 \\ 0.003316 & 0.005083 & 0.999982 & 643.361 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3\text{-}38)$$

Calculating positions of the inverts

Substituting F_{b1}^n into Equation (3-27), we get $\hat{n}_n$ and $\hat{o}_n$ as Equation (3-33).

$$\begin{cases} \hat{n}_{n1} = \begin{pmatrix} -0.013471 & 0.999906 & -0.002335 \end{pmatrix}^T \\ \hat{o}_{n1} = \begin{pmatrix} 27741.028 & 5934807.793 & 643.350 \end{pmatrix}^T \end{cases} \quad (3\text{-}39)$$

Substituting $\hat{n}_{n1}$, $\hat{o}_{n1}$, and $\hat{u}_n$ [$(0 \quad 0 \quad 1)^T$] to Equation (3-23), Equation (3-24), and Equation (3-25), we can get $\hat{h}_{n1}$, $\hat{s}_{n1}$, and $\hat{i}_{n1}$ as Equation (3-34).

$$\begin{cases} \hat{h}_{n1} = \begin{pmatrix} -0.999909 & -0.013471 & 0 \end{pmatrix}^T \\ \hat{s}_{n1} = \begin{pmatrix} 0.000031 & -0.002334 & -0.999997 \end{pmatrix}^T \\ \hat{i}_{n1} = \begin{pmatrix} 27741.028 & 5934807.790 & 642.180 \end{pmatrix}^T \end{cases} \quad (3\text{-}40)$$

Similarly, the invert coordinates at t2 can also be determined as Equation (3-35).

$$\hat{i}_{n2} = \begin{pmatrix} 27741.046 & 5934806385 & 642.179 \end{pmatrix}^T \quad (3\text{-}41)$$

CHAPTER 4: ARCHITECTURE AND TECHNICAL DETAILS OF THE VISUALIZATION SYSTEM

4.1 System Architecture

The overall architecture for the proposed visualization system is shown in Figure 4-1. It mainly consists of two components, namely, a MySQL database and a visualization program built on Microsoft XNA Framework. MySQL is a popular open-source relational database management system which supports multi-user concurrent access, and Microsoft XNA is a light-weight yet powerful engine for game development.

To model the as-built tunnel and its surrounding environment in real time, I divide data into two categories, namely, static data and time dependent data. Static data are design parameters derived from construction drawings and geotechnical reports, which include the as-designed tunnel alignment, soil layers, the ground, coordinates of prisms in the b-frame, and other related information. These data are inputted to the database manually. Time dependent data refer to TBM tracking data, which are sourced from the VLTB TBM Guidance System and are autonomously inputted by a data feed program.

With all the related information stored in the Database, the visualization program reads data from it and conducts spatial and visualization analysis. It then creates models of the as-built tunnel, the as-designed tunnel, and the surrounding environment, and it finally draws the visualization output on the screen. The key difference between our proposed visualization system and other tunnel information systems or tunnel visualization systems, is that all the models in our system are generated automatically based on parameters stored in the database. In other words, there is no need to manually draw CAD models and import these models to the visualization system.

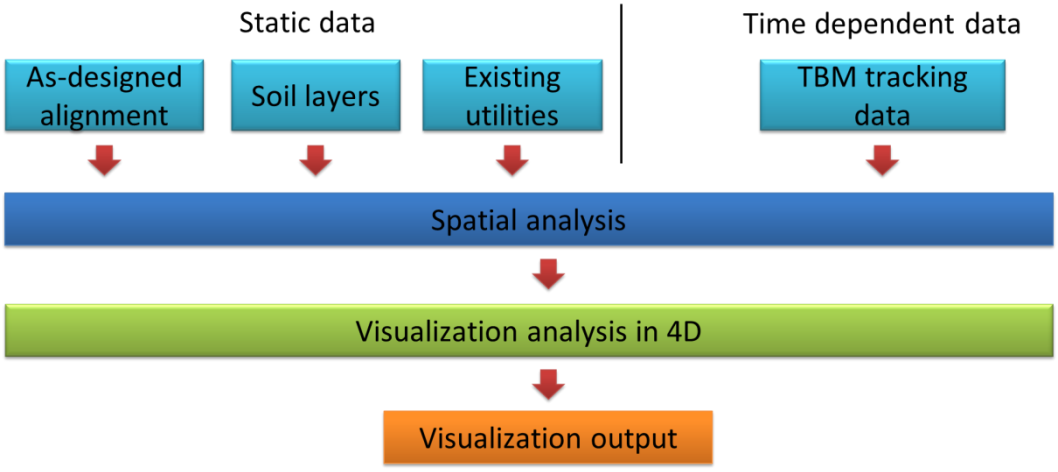


Figure 4-1: Architecture of the visualization system

4.2 Database Structure

In the MySQL database, different parameters are stored in different tables, as shown in Figure 4-2.

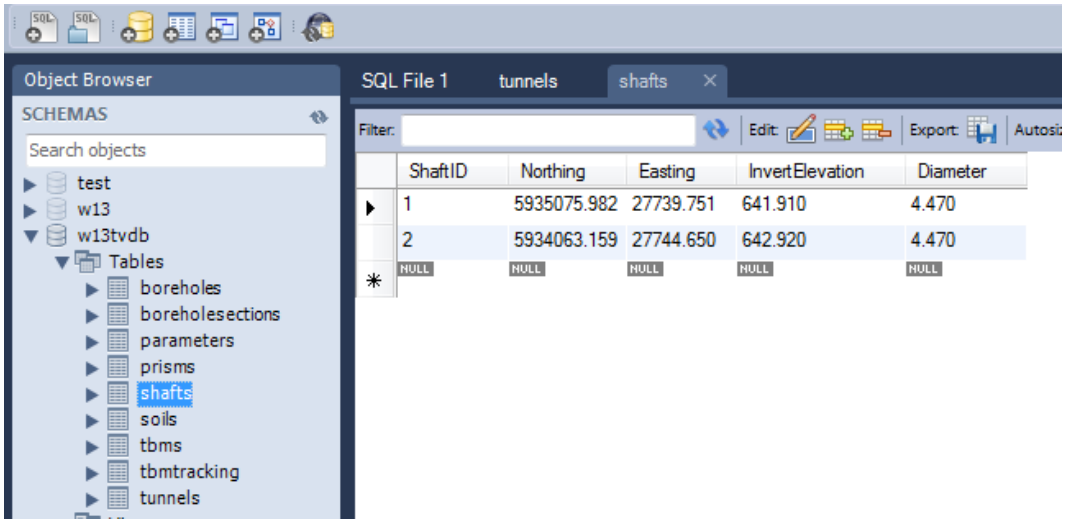


Figure 4-2: Tables in the database

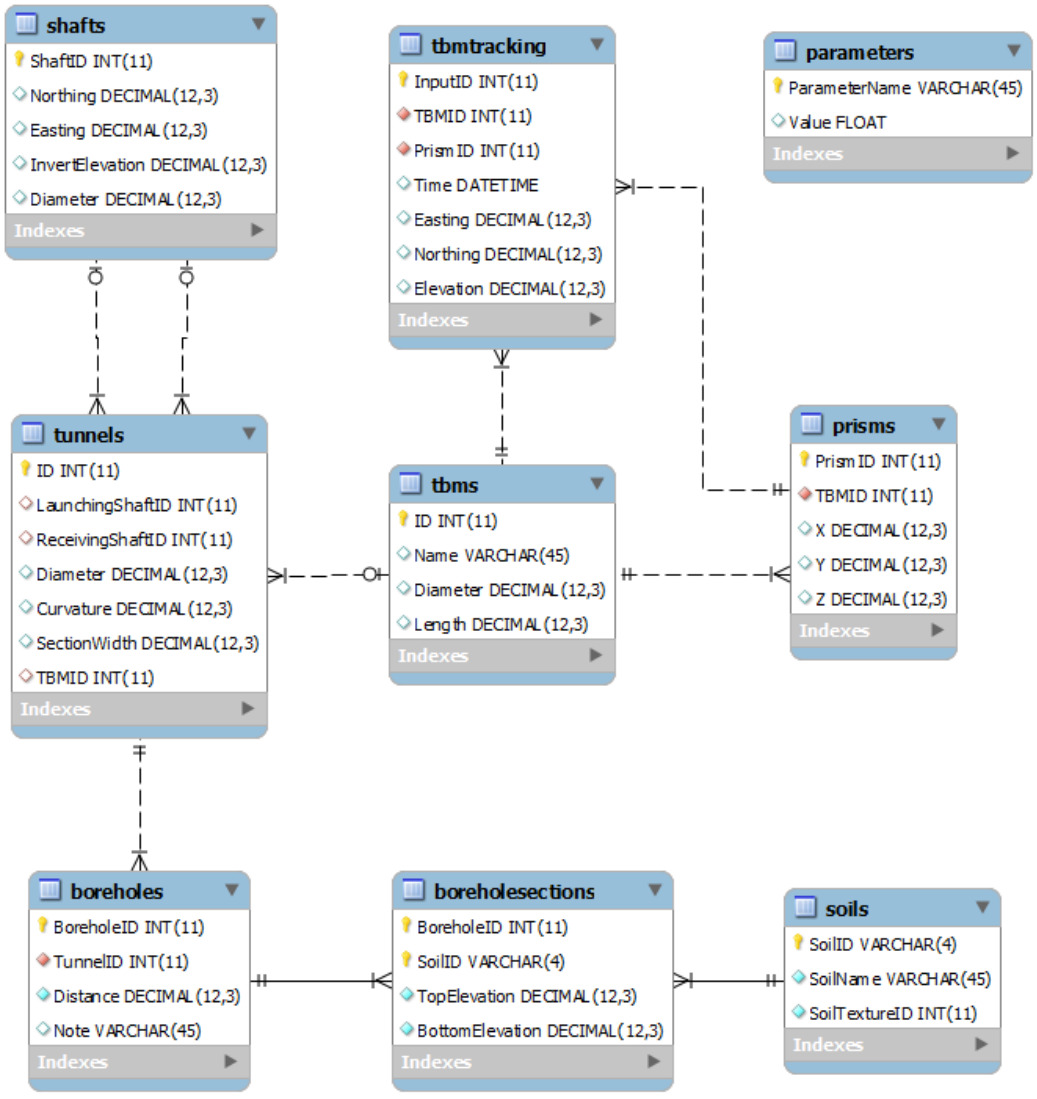


Figure 4-3: Enhanced entity–relationship model of the database

The Enhanced Entity–Relationship (EER) model of the database is shown in Figure 4-3. Due to space limit, details of these tables are not included in the thesis. With this well-defined database model, queries such as select, update or insert can be easily conducted. As the MySQL database system supports multi-user concurrent access, the data feed program can insert new TBM positioning data to the database without causing any conflict to the visualization program. As soon as a new set of positioning data is inputted

to the database, the visualization program will automatically read this new positioning data set and update the as-built tunnel model.

4.3 Visualization of Soil Layers, Tunnel, and Related Information

4.3.1 Visualization of Soil Layers

Figure 4-4 is a sample geological drawing from a tunnel project of City of Edmonton, in which we can see several soil layers interpolated from geological data in the boreholes. Typically, these boreholes are drilled along the as-designed tunnel alignment before the construction of a tunnel project.

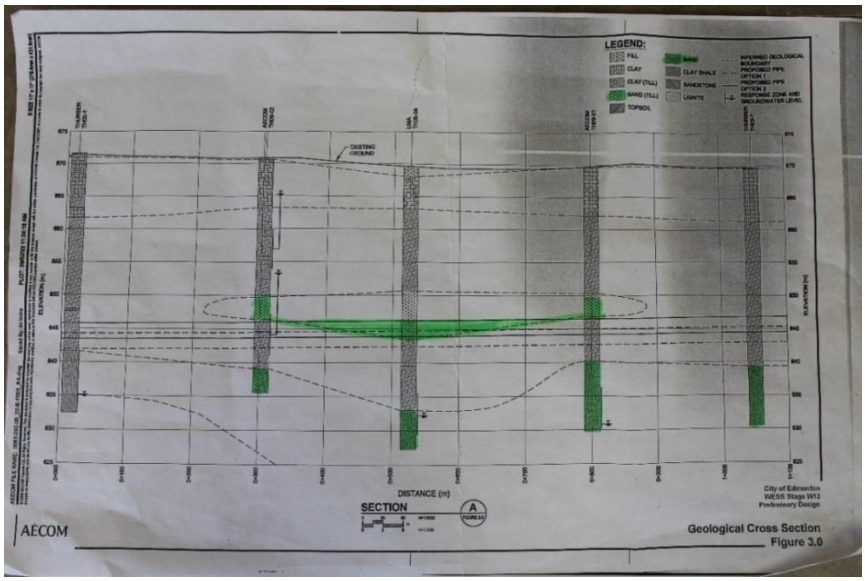


Figure 4-4: A sample geological drawing (Courtesy: City of Edmonton)

To enhance the visualization effect, it is desirable to interpolate and visualize surrounding soil layers along the tunnel alignment, like the geological drawing in Figure 4-4. In the field of geology and geoscience, several interpolation or modeling techniques, like NURBS (Non-uniform Rational B-spline) and TIN (Triangulated Irregular Network), have been used for 3D modeling of geologic bodies (Zhong et al., 2006; Tonini et al.,

2008). With a large amount of borehole data, these techniques can accurately model geologic bodies in the 3D space. However, for a typical drainage tunnel project, the boreholes are usually drilled along a straight line and the number of boreholes is still limited, which makes it inappropriate to apply those established 3D modeling methods.

To solve this problem, firstly, the vertical cross section of the ground is divided into several congruent, long and narrow rectangles. On the boundaries between two rectangles, Catmull–Rom splines are applied to interpolate soil layers; within each rectangle, it is assumed that the boundaries between different soil layers are linear. Thus, on the vertical cross section, boundaries between soil layers can be represented by several groups of line segments, as shown in Figure 4-5.

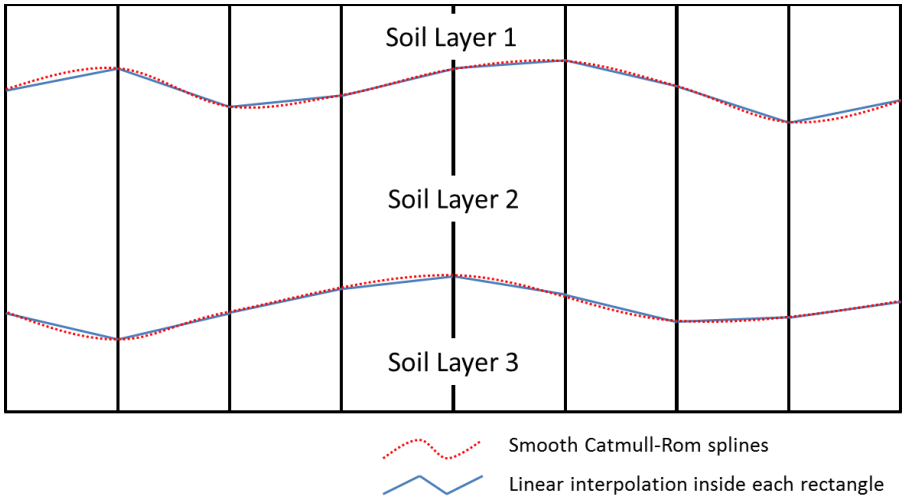


Figure 4-5: Soil layer boundary interpolation

After soil layer boundaries are interpolated from borehole data, the boundaries and rectangles are then passed to shaders (a shader is a shading program running on a graphics processing unit). To render different soil layers, multitexturing is applied, as shown in Figure 4-6, where Texture 1, Texture 2, and Texture 3 are used to render Soil Layer 1, Soil Layer 2, and Soil Layer 3, respectively.

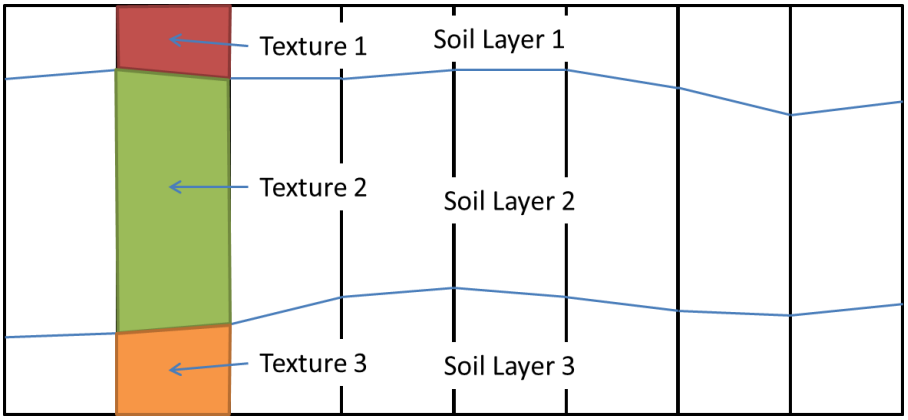


Figure 4-6: Illustration of soil layer visualization

4.3.2 Cutting the Soil

Just like a real TBM would excavate soil as it moves forward, it is also necessary to "cut" corresponding soil objects on the screen when we update the as-built tunnel model.

One alternative for cutting the soil objects is directly splitting a rectangle into two polygons, as shown in Figure 4-7. At the first glance, this approach appears to be straight forward, but it is actually difficult to be achieved, as it requires a lot of computing power to insert new vertices to existing geometries. Instead, Alpha Blending, a technique used to display an image with transparent or semi-transparent pixels (Microsoft 2013), is applied to cut the soil virtually, as shown in Figure 4-8, in which the white part represents soil excavated by the TBM.

The process of virtually cutting soil by Alpha Blending can be summarized as follows:

1. Based on the position of the as-built tunnel, determine which portions of soil layers are inside the tunnel.

2. In front of the soil layers, generate several transparent polygons (with an alpha value of 0) which covers exactly the space of excavated soil. These polygons are called soil cutting objects.
3. Draw soil cutting objects.
4. Draw soil objects.

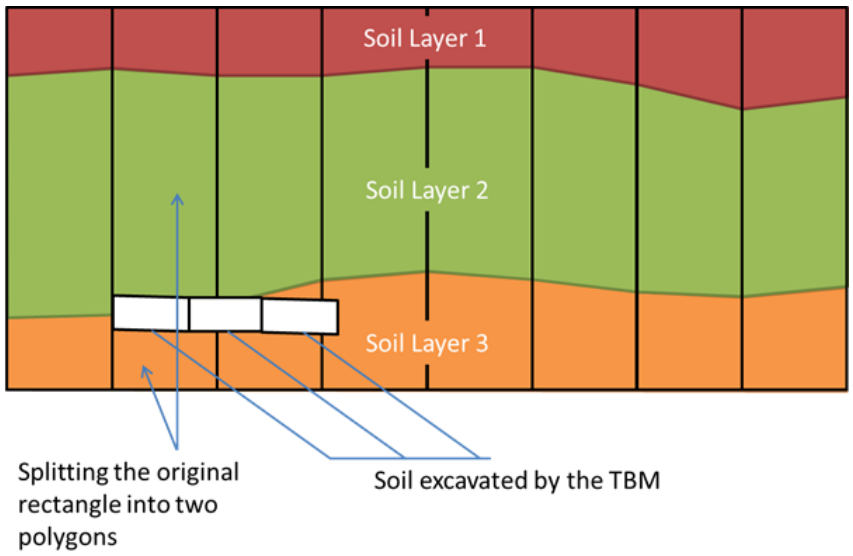


Figure 4-7: Cutting soil by splitting the rectangles

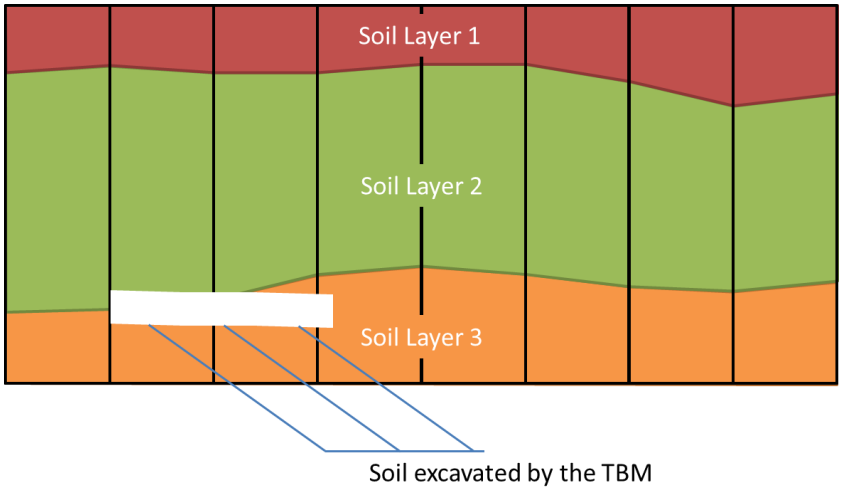


Figure 4-8: Virtually cutting soil by Alpha Blending

It is important that soil cutting objects are drawn and laid before soil objects. As such, when the game engine draws soil objects, the particular portions of soil covered by soil cutting objects will be automatically set to transparent. The result of this soil cutting algorithm is illustrated in Figure 4-9.

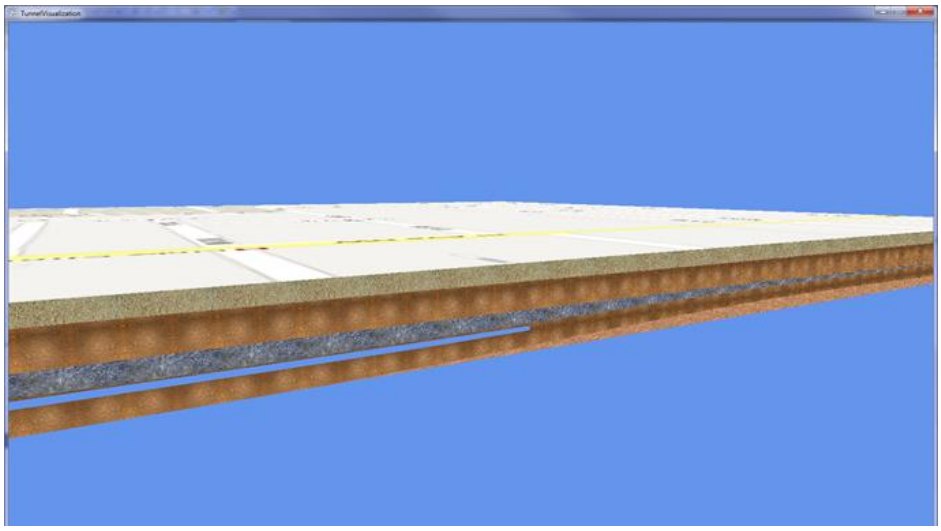


Figure 4-9: Effects of soil cutting

4.3.3 Visualization of the As-built Tunnel

After building the wire frame model in Chapter 3, textures and texture coordinates are assigned to it. Then the model is passed to shaders to generate the final visualization output, which will be presented in the next chapter.

CHAPTER 5: CASE STUDY

5.1 Project Description

To test the VLTB TBM Guidance System and the proposed as-built invert modeling and visualization approach, a drainage tunnel project of the City of Edmonton, WESS Stage W13, was chosen as the test bed. The total length, grade and inner diameter of the tunnel are 1012.6 m, 0.1%, and 2340 mm, respectively. A 4.652 m long TBM with an outer diameter of 2.526 m and was used to build this tunnel.

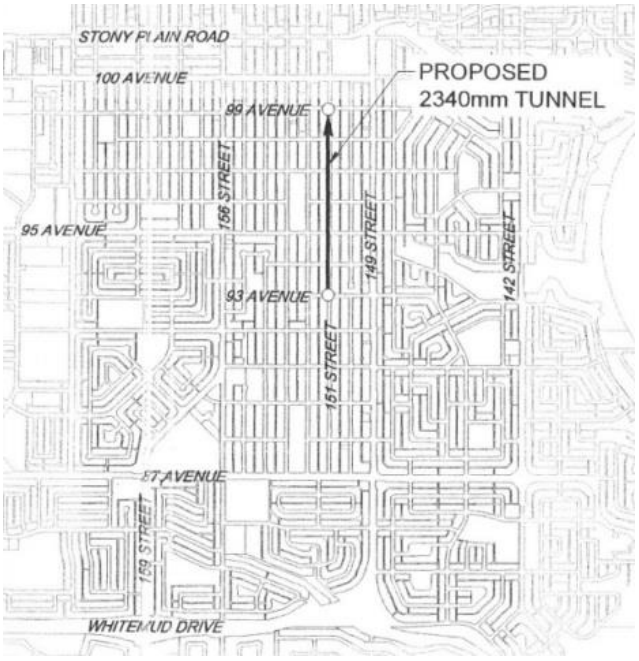


Figure 5-1: W13 Project Layout (Courtesy: AECOM)

5.2 TBM Guidance Systems

To minimize risks of guidance errors, in this project, the contractor still relied on a traditional laser guidance system as the primary tool for positioning the TBM, while the VLTB TBM guidance system was tested for validation purpose only. The selected results

of our field testing are listed in Table 5-1, revealing that the differences between these two guidance systems are acceptable, and performances of the VLTB guidance system are reliable. Note: (1) results from the laser system do not represent the true deviations (30-40mm errors according to experienced tunnel surveyors) but provide reliable benchmarks for cross checking VLTB results; (2) the VLTB guidance system integrates a robotic total station to realize high-precision point surveying at 2-3 mm accuracy.

Figure 5-2: VLTB system under testing

Table 5-1: Results of field testing (unit: mm)

Date	VLTB		Laser		Difference	
	LD	GD	LD	GD	LD	GD
10/08/2012	-25	1	-5	-20	-20	21
24/08/2012	6	15	5	15	1	0
30/08/2012	-46	-17	-5	-15	-41	-2
13/09/2012	7	-63	15	-30	-8	-33
21/09/2012	-6	-48	0	-20	-6	-28
26/09/2012	-2	-32	6	15	-8	-47
03/10/2012	-9	-21	10	20	-19	-41
21/11/2012	15	-18	0	-40	15	22

Note: LD stands for Line Deviation and GD stands for Grade Deviation.

5.3 Data Input and Transmission

Before the first field test, the design information of the W13 project, such as the as-designed alignment, borehole information, is manually entered into a database. In each field test, new TBM positioning data are automatically inputted from the robotic total station to a laptop in the trailer office above ground via ZigBee wireless sensor networks, as shown in Figure 5-3. Due to the extremely limited space constraint inside the tunnel, the VLTB TBM Guidance System can guarantee the visibility of only one prism (instead of three as desired by VLTB in its ideal application setting) in this tunnel. As input data are so limited, it is reasonably assumed that the advancing direction of the TBM is always parallel to the as-designed alignment, and the roll angle of the machine is zero.

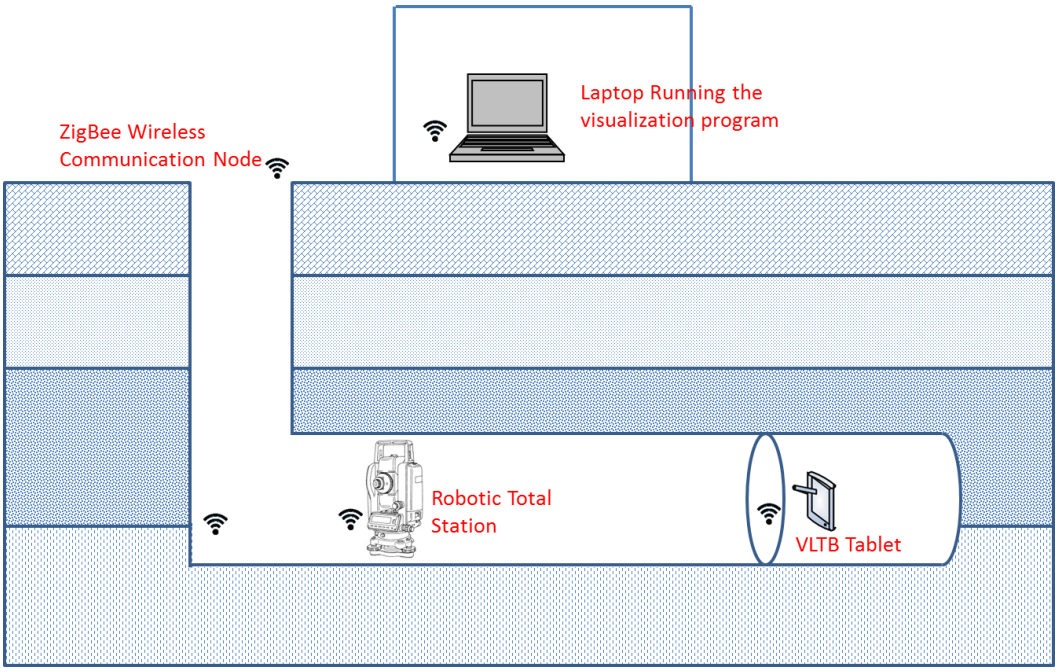


Figure 5-3: Data transmission from the total station to the trailer office

5.4 Visualization Output

A series of visualization outputs are illustrated from Figure 5-4 to Figure 5-7. As new positioning data are inputted, the as-built tunnel model is also updated on the fly.

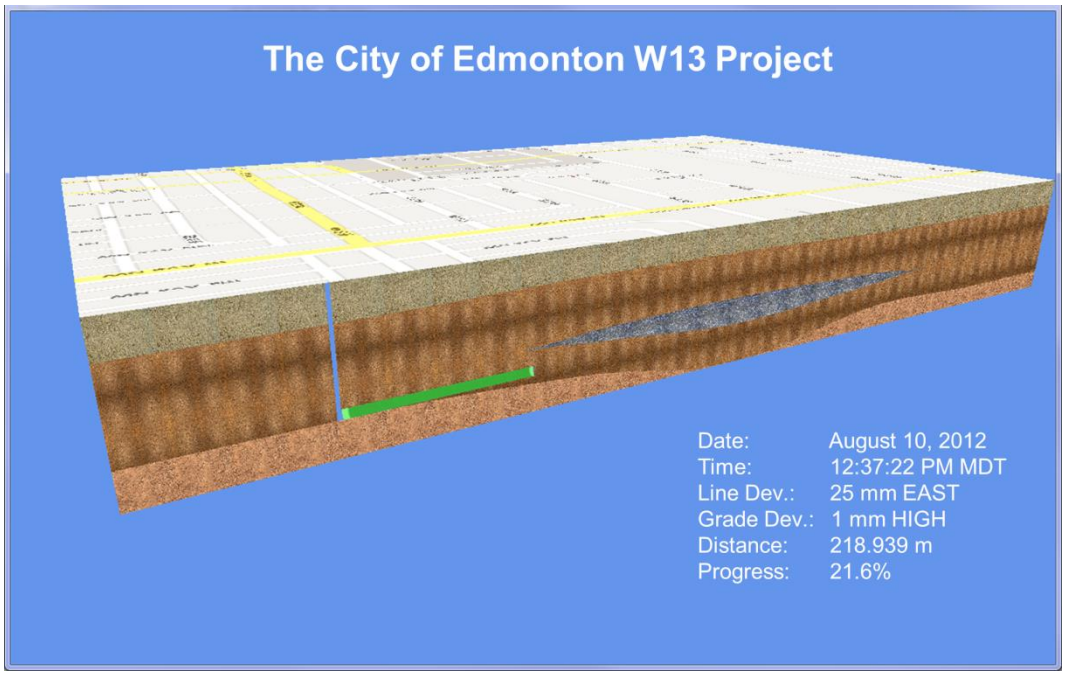


Figure 5-4: Visualization Output (August 10, 2012)

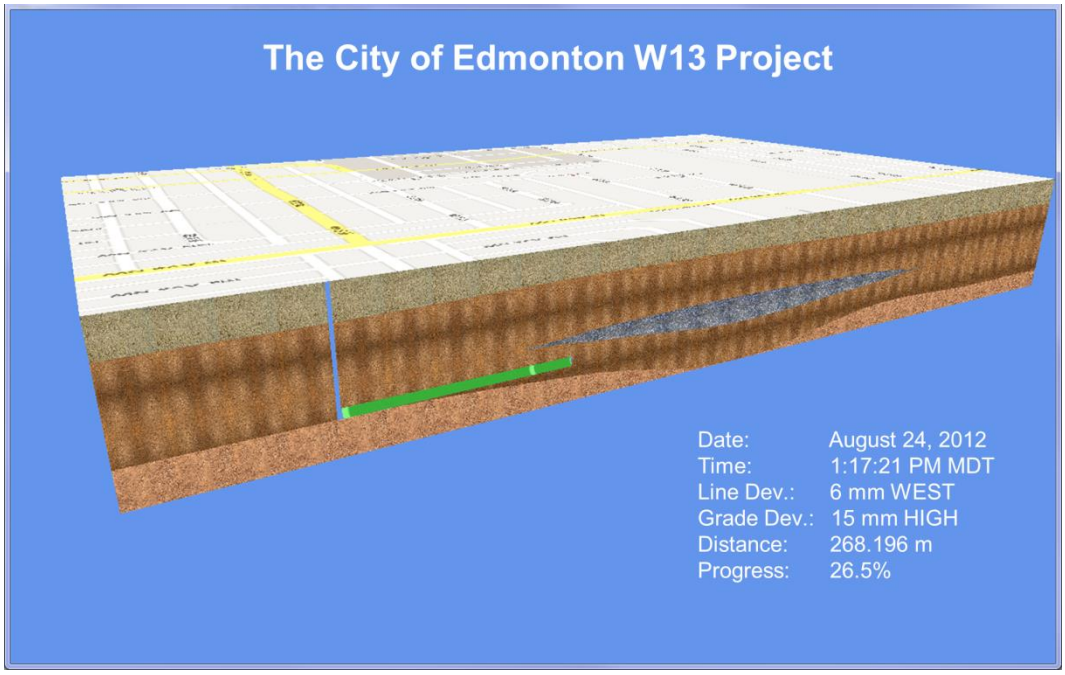


Figure 5-5: Visualization output (August 24, 2012)

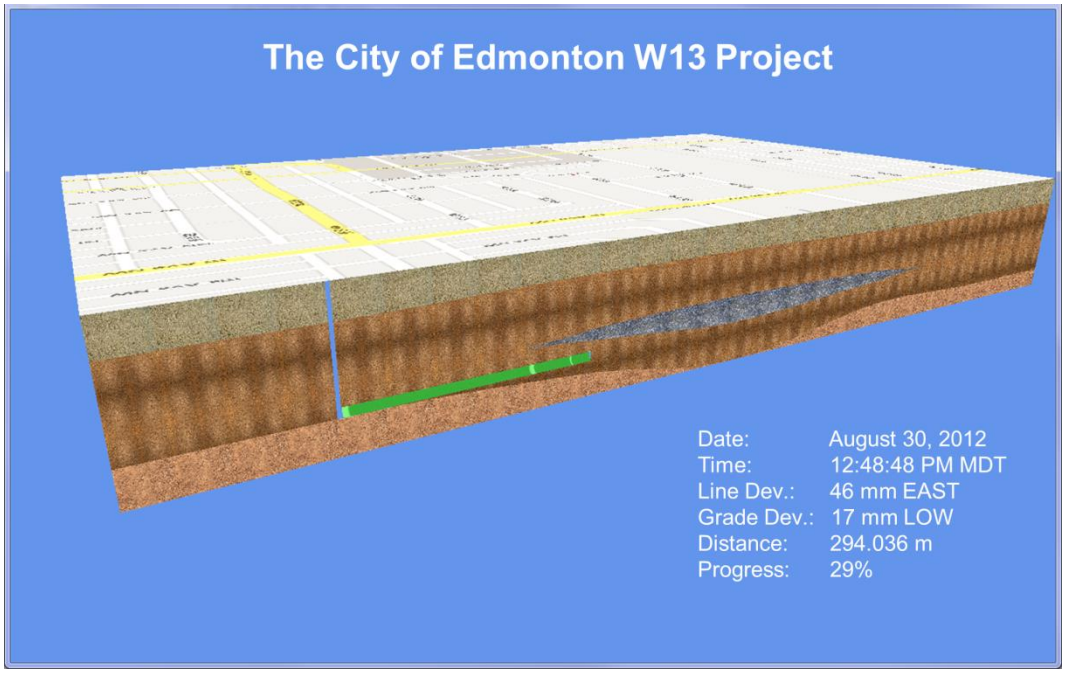


Figure 5-6: Visualization output (August 30, 2012)

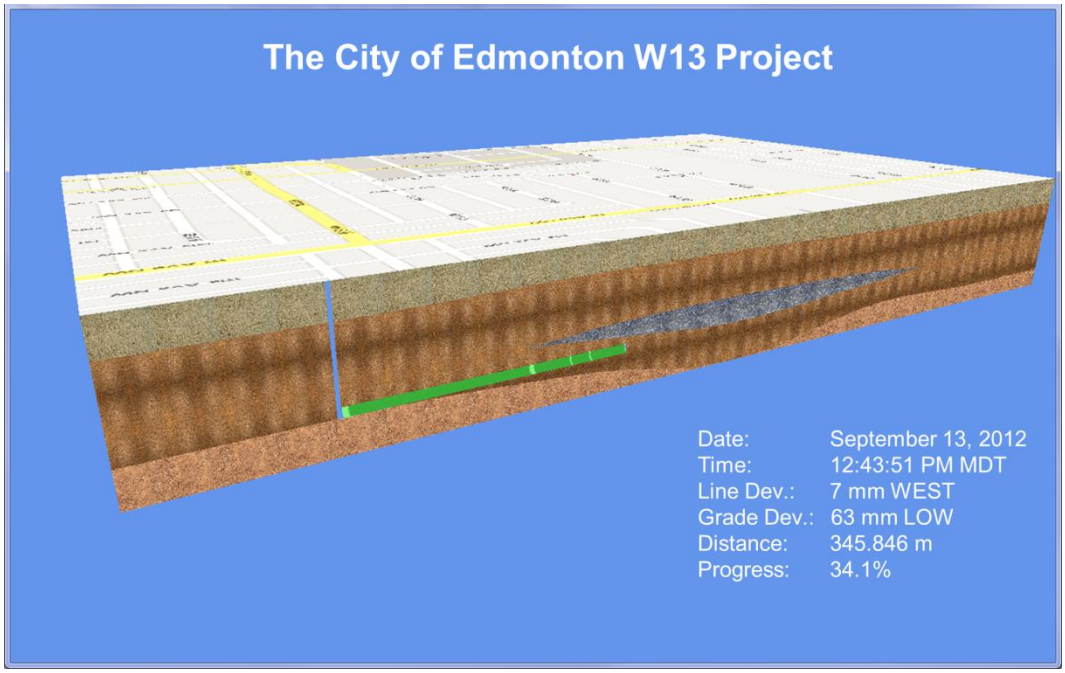


Figure 5-7: Visualization output (September 13, 2012)

5.5 Error Analysis and Validation

5.5.1 Error Classification

To validate the proposed as-built invert modeling approach, firstly, causes of errors are identified, as shown in Figure 5-8.

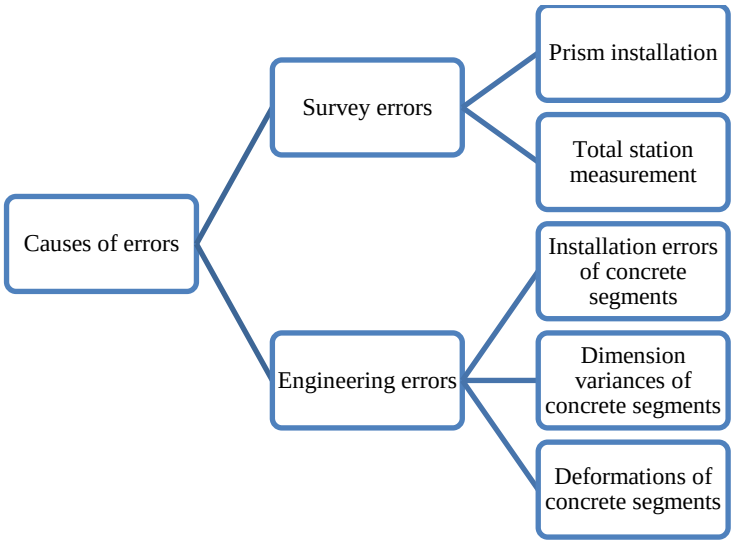


Figure 5-8: Causes of errors

The errors can be classified into two categories, namely: survey errors and engineering errors. Survey errors refer to the errors of the coordinates of prisms in both the n-frame and b-frame, such as the measurement errors of the total station and the installation errors of the prisms in the TBM. Engineering errors refer to other factors that may affect the shape and position of the as-built tunnel, such as the lining installation errors, the dimensional variances and deformations of concrete segments.

Due to limited data availability, in the current research, our focus is to quantify the total station measurement errors. Accuracy and reliability of the proposed computing approach for tunnel as-built invert modeling are evaluated by Monte Carlo Simulation based on the

known errors of the total station provided by the manufacturer (Leica), as illustrated in the following sections.

5.5.2 Total Station Measurements and Errors

When we use a total station to measure the positions of other objects, the first step is usually to initialize the coordinates of the total station in the n-frame, which is also known as the resection process. Suppose the true coordinates of a total station in the n-frame are (x_0, y_0, z_0). The position of a tracking point i can be determined by three parameters relative to the position of the total station, namely, (1) the slope distance d; (2) the horizontal angle α; and (3) the vertical angle β (Shen, Lu, and Chen 2011), as shown in Figure 5-9.

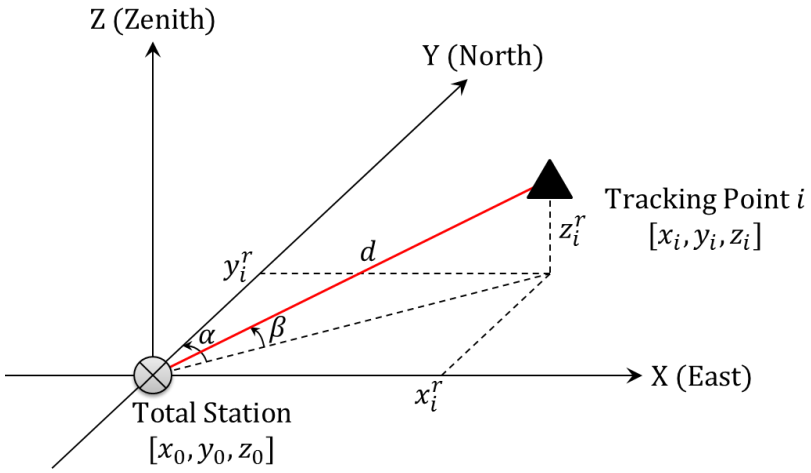


Figure 5-9: Measurements taken by a total station [adapted from Shen, Lu, and Chen (2011)]

Suppose the three parameters d, α, and β are all true values. The true coordinates of the tracking point i, relative to the total station, can be calculated by Equation (5-1):

$$\begin{cases} x_i^r = d \times \cos\beta \times \sin\alpha \\ y_i^r = d \times \cos\beta \times \cos\alpha \\ z_i^r = d \times \sin\beta \end{cases} \quad (5\text{-}1)$$

The true coordinates of point i in the n-frame, are determined by Equation (5-2):

$$\begin{cases} x_i = x_i^r + x_0 \\ y_i = y_i^r + y_0 \\ z_i = z_i^r + z_0 \end{cases} \quad (5\text{-}2)$$

However, in reality, it is impossible to get the true values of (x_0, y_0, z_0), nor the true values of d, α, and β. When we use a total station to survey a target point, errors are inevitable. Based on checking the technical specifications of a total station (Leica Geosystems 2013) and the manual survey records from the W13 project, error terms between true values and surveyed values are characterized as normal distributions as listed in Table 5-2.

Table 5-2: Comparisons between true values and surveyed values

True value	Surveyed value	Distribution of the error term
x_0	$x_0 + \Delta x_0$	Normal(0, 5mm)
y_0	$y_0 + \Delta y_0$	Normal(0, 5mm)
z_0	$z_0 + \Delta z_0$	Normal(0, 5mm)
d	$d + \Delta d$	Normal(0, 1mm+1.5 ppm)
α	$\alpha + \Delta\alpha$	Normal(0, 1'')
β	$\beta + \Delta\beta$	Normal(0, 1'')

Note:

1. Normal (x, y) denotes a normal distribution with a mean x and a standard deviation y.

2. ppm stands for parts per million. Here in Table 5-2, 1.5 ppm means that for every 1000 m distance between the total station and the object being surveyed, the standard deviation of distance measurement is increased by 1.5 mm.

5.5.3 Simulating Total Station Measurements

Suppose the true coordinates of the total station and the three prisms installed in the TBM in the tunnel are known, as listed in Table 5-3. Note the coordinates of P1, P2 and P3 are the same as those in Table 3-3 used in the previous numerical example.

Table 5-3: True coordinates of the total station and prisms (unit: m)

Point	Easting	Northing	Elevation
Total Station	27740.564	5934907.978	643.248
P1	27740.618	5934807.717	644.163
P2	27740.332	5934807.712	644.099
P3	27740.231	5934807.712	643.823

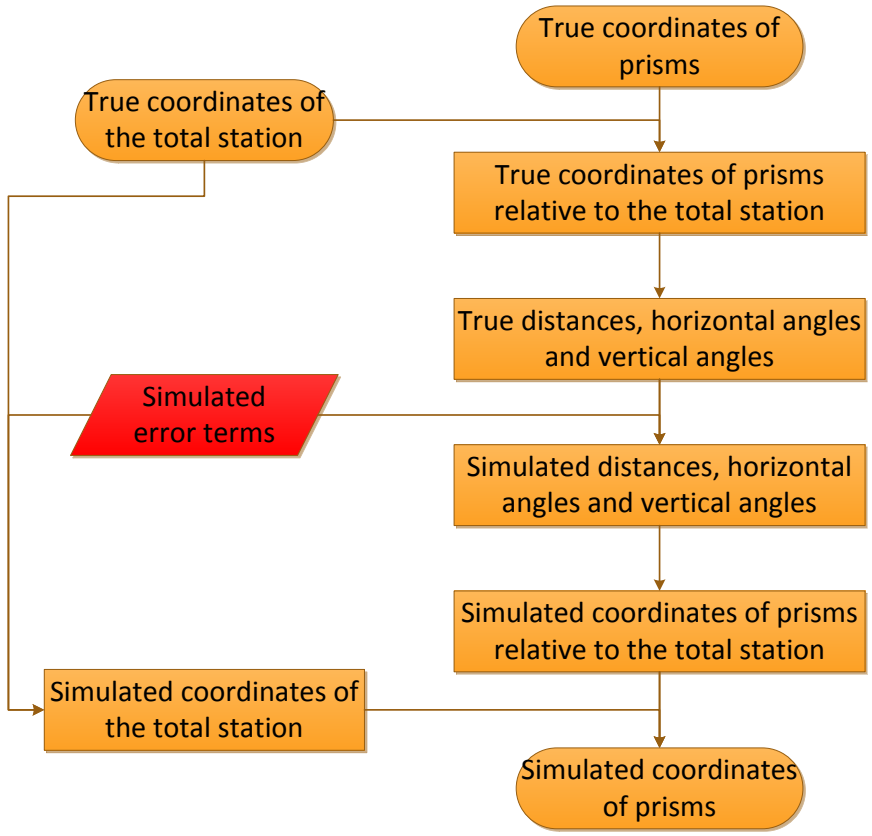


Figure 5-10: Process for simulating total station measurements

Based on the true coordinates of the total station and the prisms in Table 5-3 and the distributions of error terms in Table 5-2, relevant measurements by the total station can be simulated as follows (also illustrated in Figure 5-10). For simplicity, we assume that

the total station is perfectly leveled and the north direction of the n-frame is also perfectly registered to the total station.

Firstly, the coordinates in Table 5-3 are substituted to Equation (5-2) and Equation (5-1), and we can obtain the true coordinates of prisms relative to the total station (x^r, y^r, and z^r) and the true distances, horizontal angles, and vertical angles (d, α, and β), for P1, P2, and P3, as listed in Table 5-4.

Table 5-4: True values for P1, P2, and P3

Point	x^r (m)	y^r (m)	z^r (m)	d (m)	α (rad)	β (rad)
P1	0.054	-100.261	0.915	100.265	3.141054	0.009126
P2	-0.232	-100.266	0.851	100.270	3.143906	0.008487
P3	-0.333	-100.266	0.575	100.268	3.144914	0.005735

Next, error terms are simulated based on the distributions in Table 5-2. By adding the simulated error terms to the true distances, horizontal angles, and vertical angles, the simulated total station measurements of distances, horizontal angles, and vertical angles can be determined. Substituting these values to Equation (5-1), we can get the simulated coordinates of prisms relative to the total station.

Finally, the coordinates of the total station are simulated by adding the simulated error terms to the true coordinates. Substituting the simulated coordinates of prisms relative to the total station and the simulated coordinates of the total station to Equation (5-2), we get the simulated coordinates of prisms in the n-frame.

5.5.4 Simulating As-built Modeling Errors from Total Station Measurements

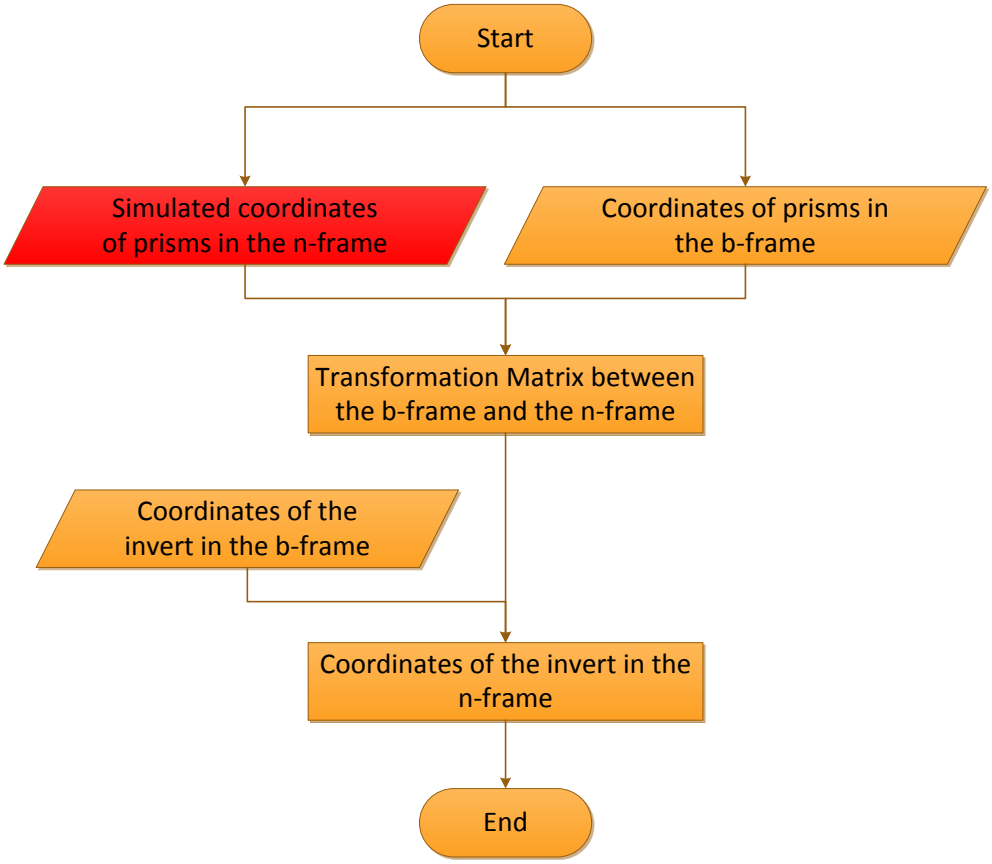


Figure 5-11: Monte Carlo Simulation process

Based on the simulated coordinates of the prisms in the n-frame from the last section, as-built modeling errors resulting from the effects of total station measurements can be evaluated by Monte Carlo Simulation, as shown in Figure 5-11. The process of one single run of the Monte Carlo Simulation is similar to that of the numeric example in Chapter 3, except that input data here are not deterministic but randomly sampled.

By repetitively running Monte Carlo Simulation for 10000 times, and comparing the results with the true coordinates in Chapter 3, the errors can be summarized as Table 5-5, Figure 5-12, Figure 5-13, and Figure 5-14.

Table 5-5: Summary of errors (unit: mm)

	Easting	Northing	Elevation
Standard Deviation	5.4	19.0	5.2
90% Confidence Interval	±9	±32	±9

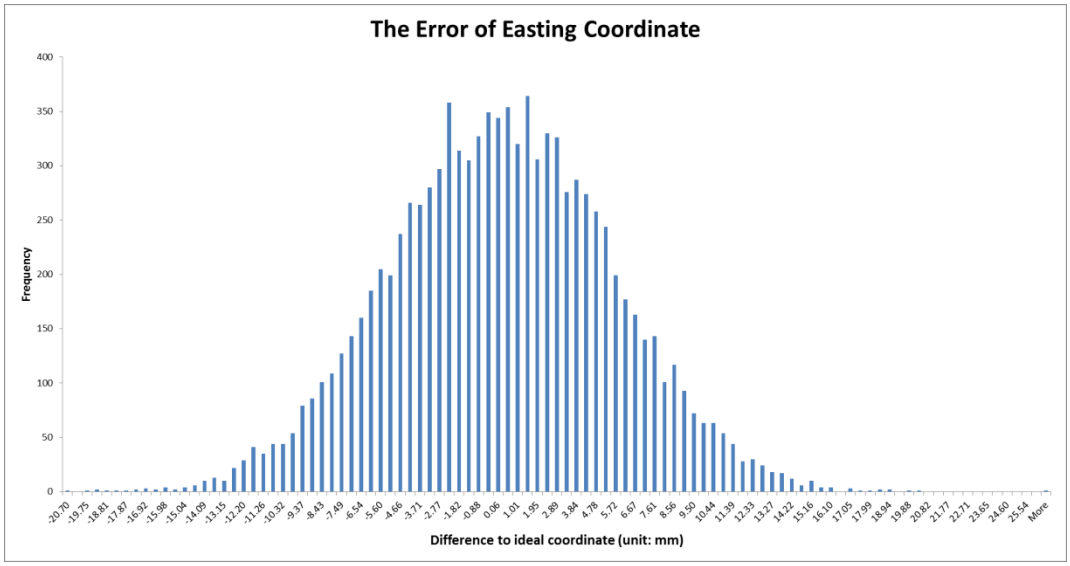


Figure 5-12: The error distribution of easting coordinate

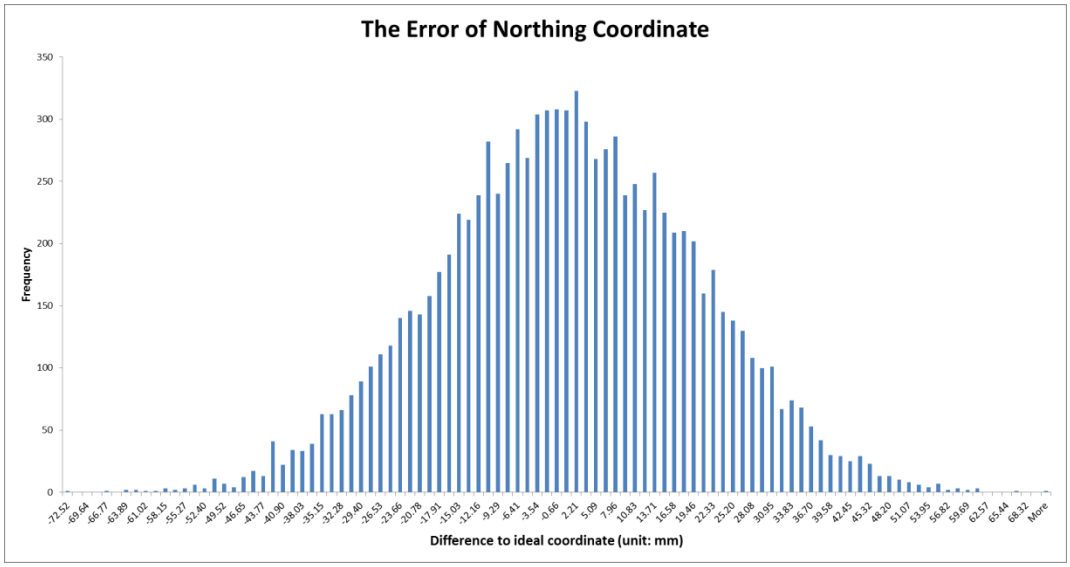


Figure 5-13: The error distribution of northing coordinate

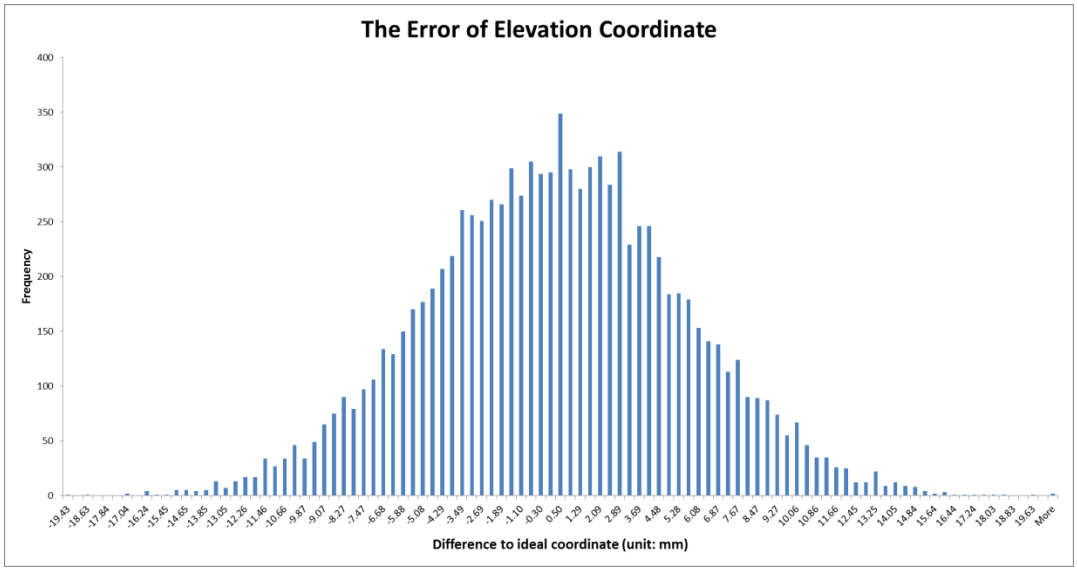


Figure 5-14: The error distribution of elevation coordinate

Note the error on the nothing direction is higher than those on the other two directions (easting and elevation). As denoted by Shen, Lu, and Chen (2011), the accuracy of the three orientation angles derived from the point-to-angle algorithm is dependent on the geometric layout of the prisms installed on the TBM. As we can see in Figure 3-5, the Prism P2 is too close to the line segment P1-P3, which results in larger errors on the yaw and pitch angles, thus affecting the modeling accuracy on the northing direction. Nonetheless, the resultant northing errors are characterized around 2 cm in the current demanding constraint of prism layout design being simulated, which reflects the adverse visibility scenario potentially encountered in practical tunnel construction. Through repetitive surveying (applying least square optimization) or laying out better geometry of the three prisms, the modeling errors can generally be reduced.

CHAPTER 6: CONCLUSIONS AND DISCUSSIONS

With the trend of global urbanization and underground space development, the underground space is becoming more and more congested, which results in increased construction risks and higher quality assurance standard in terms of the alignment and positioning of the as-built tunnel. To address these challenges, we need to find out a way, which is fast, accurate and low-cost, to map out as-built tunnels, especially the tunnel inverts. In recent years, several new technologies have been developed and applied for the as-built modeling of tunnels, such as laser scanning and GPR, but none is yet able to perform sufficiently and meet practical requirements for cost-effectiveness.

In this thesis, a new as-built tunnel invert modeling approach is proposed by using real-time TBM positioning data from a TBM Guidance System (VLTB) which has been developed from construction-survey integration and automation research. A case study in Edmonton, Alberta was conducted to demonstrate its feasibility. The advantages of this approach can be summarised as follows:

- Ability to perform as-built invert modeling during the construction stage in real-time
- As-built invert models are developed based on readily-available TBM positioning data, no extra worker, cost, or equipment is needed.
- Models are presented to engineers and project managers in a convenient and user-friendly way via an in-house developed visualization program.
- All data are stored parametrically in a database. The database and program can be easily reused for a new tunneling project.

Nonetheless, the proposed approach ignores effects of the deformation of the tunnel under ground pressure, installation errors and dimensional discrepancies of concrete lining segments, which would potentially lead to higher modeling errors. Besides, as we can see from the field test in the W13 project, in practical applications under extremely challenging scenarios, the line of sight between the total station and three target prisms installed on the TBM cannot be guaranteed. With only one prism visible, we have to make some assumptions on the orientation angles of the TBM; as a result, the modeling accuracy is also decreased.

An error analysis based on Monte Carlo simulation is conducted to quantify the accuracy of the proposed approach. To further evaluate the accuracy and reliability of the proposed approach in the near future, it is advisable to conduct a field survey of the tunnel inverts on sampled tunnel sections and cross check against the as-built invert model resulting from our proposed approach.

REFERENCES

Alberta One-Call. "Click before You Dig - Learning Center - the Process." Alberta One-Call Corporation, accessed 07/18, 2013, http://www.albertaonecall.com/learning-centre/the-process/.

Beck, A. R., G. Fu, A. G. Cohn, B. Bennett, and J. G. Stell. 2007. "A Framework for Utility Data Integration in the UK." In *Urban and Regional Data Management - Proceedings of the Urban Data Management Society Symposium 2007*, edited by V. Coors, M. Rumor, E. M. Fendel and S. Zlatanova, 261-276: Taylor and Francis.

Beck, A., A. G. Cohn, J. Parker, N. Boukhelifa, and G. Fu. 2009. "Seeing the Unseen: Delivering Integrated Underground Utility Data in the UK." In *Proceedings of the GeoWeb Conference*, Vancouver, July, 2009.

Costello, S. B., D. N. Chapman, C. D. F. Rogers, and N. Metje. 2007. "Underground Asset Location and Condition Assessment Technologies." *Tunnelling and Underground Space Technology* 22 (5–6): 524-542. doi:http://dx.doi.org/10.1016/j.tust.2007.06.001.

Duffell, C. and D. Rudrum. 2005. "Remote Sensing Techniques for Highway Earthworks Assessment." In *Site Characterization and Modeling*, 1-13: American Society of Civil Engineers. doi:doi:10.1061/40785(164)3.

Durmisevic, Sanja. 1999. "The Future of the Underground Space." *Cities* 16 (4): 233-245. doi:http://dx.doi.org/10.1016/S0264-2751(99)00022-0.

Fekete, Stephanie, Mark Diederichs, and Matthew Lato. 2010. "Geotechnical and Operational Applications for 3-Dimensional Laser Scanning in Drill and Blast Tunnels." *Tunnelling and Underground Space Technology* 25 (5): 614-628. doi:10.1016/j.tust.2010.04.008.

Ghassemi, M., D. Zoldy, and H. Javady. 2010. "Towards Precise Under Ground Mapping System in Canada." In *North American Tunneling: 2010 Proceedings*, edited by Lawrence R. Eckert, 116-121. Littleton, Colo: Society for Mining, Metallurgy, and Exploration, Inc. SME].

Grübl, Fritz. 2012. "Segmental Ring Design – New Challenges with High Tunnel Diameters." *Tunnel*, 04/2012, 10-24.

Guha, Sumanta. 2011. *Computer Graphics through OpenGL: From Theory to Experiments*. Boca Raton, FL, United States: Chapman & Hall/CRC Press.

Hapgood, Fred. 2004. "The Underground Cutting Edge: The Innovators Who made Digging Tunnels High-Tech." *Invention & Technology*, Fall 2004.

ITA. "White Paper 2 Planning the use of Underground Space." International Tunnelling and Underground Space Association, last modified May, accessed 07/17, 2013, https://ita-aites.org/fr/wg-committees/committees/itacus/downloads/159-planning-the-use-of-underground-space.

Klein, Laura, Nan Li, and Burcin Becerik-Gerber. 2012. "Imaged-Based Verification of as-Built Documentation of Operational Buildings." *Automation in Construction* 21 (0): 161-171. doi:10.1016/j.autcon.2011.05.023.

Leica Geosystems. "Leica Viva TS15 Datasheet." accessed 09/03, 2013, http://www.leica-geosystems.com/downloads123/zz/tps/Viva%20TS15/brochures-datasheet/Leica%20Viva%20TS15%20Datasheet_en.pdf.

Li, Xiaojun, Hehua Zhu, and Lu Zhen. 2009. "Digitalization and Application of Shield Tunnels." *Chinese Journal of Geotechnical Engineering* 31 (9): 1456-1461.

Li, Xiaojun and Hehua Zhu. 2009. "Modeling and Visualization of Underground Structures." *Journal of Computing in Civil Engineering* 23 (6): 348-354. doi:10.1061/(ASCE)0887-3801(2009)23:6(348).

Maidl, Bernhard, Martin Herrenknecht, Ulrich Maidl, and Gerhard Wehrmeyer. 2012. *Mechanised Shield Tunnelling*. 2nd ed. Berlin, Germany: Ernst & Sohn.

Maire, P., P. Blunier, A. Parriaux, and L. Tacher. September 21-22, 2006. "Underground Planning and Optimisation of the Underground Ressources' Combination Looking for Sustainable Development in Urban Areas." In *Going Underground: Excavating the Subterranean City*, Manchester, UK.

Megaw, T. M. and J. V. Bartlett. 1981. *Tunnels--Planning, Design, Construction*. Ellis Horwood Series in Engineering Science. Vol. 1. New York: Chichester, West Sussex : Ellis Horwood.

Metje, N., P. R. Atkins, M. J. Brennan, D. N. Chapman, H. M. Lim, J. Machell, J. M. Muggleton, et al. 2007. "Mapping the Underworld - State-of-the-Art Review." *Tunnelling and Underground Space Technology* 22 (5-6): 568-586. doi:10.1016/j.tust.2007.04.002.

Microsoft. "Alpha Blending (Direct3D 9)." last modified August 13, 2013, accessed 09/20, 2013, http://msdn.microsoft.com/en-us/library/windows/desktop/bb172251%28v=vs.85%29.aspx.

PHMSA. "All Reported Pipeline Incidents by Cause." Pipeline & Hazardous Materials Safety Administration, U.S. Department of Transportation, accessed July 17, 2013, http://primis.phmsa.dot.gov/comm/reports/safety/AllPSIDet_1993_2012_US.html?nocache=8500.

Schaefer, V., S. Burckhard, and J. Boomer. 2005. "Landslide Mapping with Airborne Laser Swath Mapping (ALSM): Accuracy Assessment." In *Site Characterization*

and Modeling, 1-12: American Society of Civil Engineers. doi:doi:10.1061/40785(164)1.

Shen, Xuesong, Ming Lu, Siri Fernando, and Simaan AbouRizk. 2012. "Tunnel Boring Machine Positioning Automation in Tunnel Construction." In *Proceedings ISARC 2012*, Eindhoven, the Netherlands, International Association for Automation and Robotics in Construction.

Shen, Xuesong, Ming Lu, and Wu Chen. 2011. "Computing Three-Axis Orientations of a Tunnel-Boring Machine through Surveying Observation Points." *Journal of Computing in Civil Engineering* 25 (3): 232-241. doi:10.1061/(ASCE)CP.1943-5487.0000087.

Shen, Xuesong, Ming Lu, and Wu Chen. 2011. "Tunnel-Boring Machine Positioning during Microtunneling Operations through Integrating Automated Data Collection with Real-Time Computing." *Journal of Construction Engineering and Management-Asce* 137 (1): 72-85. doi:10.1061/(ASCE)CO.1943-7862.0000250.

Sivan Design. "3D GIS in the Cloud." accessed 07/19, 2013, http://www.sivandesign.com/products/3dgis.

Tang, Pingbo and Burcu Akinci. 2012. "Formalization of Workflows for Extracting Bridge Surveying Goals from Laser-Scanned Data." *Automation in Construction* 22 (0): 306-319. doi:http://dx.doi.org/10.1016/j.autcon.2011.09.006.

The City of Edmonton. "Design and Construction Standards, Drainage, Volume 3." accessed 01/24, 2013, http://www.edmonton.ca/business_economy/documents/Volume_3_Drainage_.pdf.

Tng, Serene. 2012. "Going Underground." *Skyline*, Nov/Dec, 10-11.

Tonini, Andrea, Enrico Guastaldi, Giovanni Massa, and Paolo Conti. 2008. "3D Geo-Mapping Based on Surface Data for Preliminary Study of Underground Works: A Case Study in Val Topina (Central Italy)." *Engineering Geology* 99 (1-2): 61-69. doi:10.1016/j.enggeo.2008.02.010.

van Gosliga, R., R. Lindenbergh, and N. Pfeifer. 2006. "Deformation Analysis of a Bored Tunnel by Means of Terrestrial Laser Scanning." Dresden, Germany, International Society for Photogrammetry and Remote Sensing Volume XXXVI, Part 5, 25-27 September.

Young, G., M. Wallbom, S. DiBenedetto, F. Romero, and K. Sjostrom. 2009. "Advances in Underground Imaging." In *International No-Dig Show 2009*, Totonto, Canada, North American Society for Trenchless Technology.

Zhong, Deng-Hua, Ming-Chao Li, Ling-Guang Song, and Gang Wang. 2006. "Enhanced NURBS Modeling and Visualization for Large 3D Geoengineering Applications: An Example from the Jinping First-Level Hydropower Engineering Project, China." *Computers & Geosciences* 32 (9): 1270-1282. doi:10.1016/j.cageo.2005.11.007.

Zou, J., M. Lu, R. Karumudi, and X. Shen. 2012. "Application Potential of Ultra-Wide Band Radar for Detecting Buried Obstructions in Construction." In *Construction Research Congress 2012*, 868-878: American Society of Civil Engineers. doi:doi:10.1061/9780784412329.088.

APPENDIX A: HIGH-LEVEL SHADING LANGUAGE SOURCE CODE FOR SOIL CUTTING

```
// Maximum soil layers, 10
float4x4 World;
float4x4 View;
float4x4 Projection;

uniform float TextureTiling;
uniform int NofSoilLayers;
uniform float leftXPlusY;
uniform float rightXPlusY;
float leftBottomElevations [10];
float rightBottomElevations [10];
float leftTopElevations [10];
float rightTopElevations [10];

texture2D Texture0;
sampler2D TS0 = sampler_state {
        Texture = <Texture0>;
        AddressU = Wrap;
        AddressV = Wrap;
        MinFilter = Anisotropic;
        MagFilter = Anisotropic;
};

texture2D Texture1;
sampler2D TS1 = sampler_state {
```

```
        Texture = <Texture1>;
        AddressU = Wrap;
        AddressV = Wrap;
        MinFilter = Anisotropic;
        MagFilter = Anisotropic;
};

texture2D Texture2;
sampler2D TS2 = sampler_state {
        Texture = <Texture2>;
        AddressU = Wrap;
        AddressV = Wrap;
        MinFilter = Anisotropic;
        MagFilter = Anisotropic;
};

texture2D Texture3;
sampler2D TS3 = sampler_state {
        Texture = <Texture3>;
        AddressU = Wrap;
        AddressV = Wrap;
        MinFilter = Anisotropic;
        MagFilter = Anisotropic;
};

texture2D Texture4;
sampler2D TS4 = sampler_state {
        Texture = <Texture4>;
```

```
        AddressU = Wrap;
        AddressV = Wrap;
        MinFilter = Anisotropic;
        MagFilter = Anisotropic;
};

texture2D Texture5;
sampler2D TS5 = sampler_state {
        Texture = <Texture5>;
        AddressU = Wrap;
        AddressV = Wrap;
        MinFilter = Anisotropic;
        MagFilter = Anisotropic;
};

texture2D Texture6;
sampler2D TS6 = sampler_state {
        Texture = <Texture6>;
        AddressU = Wrap;
        AddressV = Wrap;
        MinFilter = Anisotropic;
        MagFilter = Anisotropic;
};

texture2D Texture7;
sampler2D TS7 = sampler_state {
        Texture = <Texture7>;
        AddressU = Wrap;
```

```
        AddressV = Wrap;
        MinFilter = Anisotropic;
        MagFilter = Anisotropic;
};

texture2D Texture8;
sampler2D TS8 = sampler_state {
        Texture = <Texture8>;
        AddressU = Wrap;
        AddressV = Wrap;
        MinFilter = Anisotropic;
        MagFilter = Anisotropic;
};

texture2D Texture9;
sampler2D TS9 = sampler_state {
        Texture = <Texture9>;
        AddressU = Wrap;
        AddressV = Wrap;
        MinFilter = Anisotropic;
        MagFilter = Anisotropic;
};

struct VertexShaderInput
{
   float4 Position : POSITION0;
        float2 UV : TEXCOORD0;
};
```

```
struct VertexShaderOutput
{
    float4 Position : POSITION0;
        float2 UV : TEXCOORD0;
        float4 WorldPosition : TEXCOORD1;
};

VertexShaderOutput VertexShaderFunction(VertexShaderInput input)
{
    VertexShaderOutput output;

    float4 worldPosition = mul(input.Position, World);
    float4 viewPosition = mul(worldPosition, View);
    output.Position = mul(viewPosition, Projection);
        output.WorldPosition = worldPosition;
        output.UV = input.UV;
    return output;
}

float4 PixelShaderFunction(VertexShaderOutput input) : COLOR0
{
    // TODO: add your pixel shader code here.
        float3 tex[10];
        tex[0] = tex2D(TS0, input.UV * TextureTiling);
        tex[1] = tex2D(TS1, input.UV * TextureTiling);
        tex[2] = tex2D(TS2, input.UV * TextureTiling);
        tex[3] = tex2D(TS3, input.UV * TextureTiling);
```

```
	tex[4] = tex2D(TS4, input.UV * TextureTiling);
	tex[5] = tex2D(TS5, input.UV * TextureTiling);
	tex[6] = tex2D(TS6, input.UV * TextureTiling);
	tex[7] = tex2D(TS7, input.UV * TextureTiling);
	tex[8] = tex2D(TS8, input.UV * TextureTiling);
	tex[9] = tex2D(TS9, input.UV * TextureTiling);

	float XPlusY = input.WorldPosition[0] + input.WorldPosition[1];
	float amt = (XPlusY - leftXPlusY) / (rightXPlusY - leftXPlusY);
	if(amt < 0)
	{
		amt = -amt;
	}

	float3 returnedTex = tex[0];
	for (int i = 0; i < NofSoilLayers; i++)
	{
		float BottomElevation = amt * rightBottomElevations[i] + (1 - amt) * leftBottomElevations[i];
		float TopElevation = amt * rightTopElevations[i] + (1 - amt) * leftTopElevations[i];
		if(input.WorldPosition[2] > BottomElevation && input.WorldPosition[2] < TopElevation)
		{
			returnedTex = tex[i];
		}
	}
   return float4(returnedTex, 1);
}
```

```
technique Technique1
{
    pass Pass1
    {
        VertexShader = compile vs_3_0 VertexShaderFunction();
        PixelShader = compile ps_3_0 PixelShaderFunction();
    }
}
```

APPENDIX B: C++ SOURCE CODE FOR STATISTICAL ANALYSIS

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <Eigen/Dense>
#include <cmath>
#include <random>

using namespace Eigen;
using namespace std;

#define PI 3.14159265358979323846264338327950
#define SecondsPerPi 648000.0

Matrix4d threePointstoFrame (Vector4d * p1, Vector4d * p2, Vector4d * p3)
{
	// Generate a Frame based on the positions of three points
	Vector4d v14d = *p2 - *p1;
	Vector4d v24d = *p3 - *p1;
	Vector3d v1; v1 << v14d.head(3);
	Vector3d v2; v2 << v24d.head(3);

	Vector3d r1 = v1; r1.normalize();
	Vector3d r2 = v1.cross(v2); r2.normalize();
	Vector3d r3 = r1.cross(r2);
```

```
	MatrixXd F(4,4);
	F.block(0, 0, 3, 3) << r1, r2, r3;
	F.block(0, 3, 4, 1) << *p1;
	F.block(3, 0, 1, 3) << 0.0, 0.0, 0.0;

	return F;
}

Matrix4d bodyFrameToBodyFramewithoutRolling (Matrix4d * Fnb)
{
	Vector3d y; y << (*Fnb).block(0, 1, 3, 1);
	Vector3d z_o(0, 0, 1);
	Vector3d x = y.cross(z_o);
	x.normalize();
	Vector3d z = x.cross(y);

	MatrixXd Fnb_wr(4, 4);
	Fnb_wr.block(0, 0, 3, 3) << x, y, z;
	Fnb_wr.block(0, 3, 4, 1) << (*Fnb).block(0, 3, 4, 1);
	Fnb_wr.block(3, 0, 1, 3) << 0.0, 0.0, 0.0;

	return Fnb_wr;
}

double totalStationPrismToHrztAngle(Vector3d * totalStation, Vector4d * pn)
{
	double alpha = 0.0;
	double xia = (*pn)(0) - (*totalStation)(0);
```

```
	double yia = (*pn)(1) - (*totalStation)(1);
	double zia = (*pn)(2) - (*totalStation)(2);
	double tanAlpha = xia/yia;
	if(yia >= 0)
	{
		alpha = atan(tanAlpha);
	}
	else
	{
		alpha = PI + atan(tanAlpha);
	}
	return alpha;
}

double totalStationPrismToVtclAngle(Vector3d * totalStation, Vector4d * pn)
{
	double beta = 0.0;
	double xia = (*pn)(0) - (*totalStation)(0);
	double yia = (*pn)(1) - (*totalStation)(1);
	double zia = (*pn)(2) - (*totalStation)(2);
	double tanBeta = zia/sqrt(xia*xia + yia*yia);
	beta = atan(tanBeta);
	return beta;
}

double totalStationPrismToDistance(Vector3d * totalStation, Vector4d * pn)
{
	double distance = 0.0;
```

```
	double xia = (*pn)(0) - (*totalStation)(0);
	double yia = (*pn)(1) - (*totalStation)(1);
	double zia = (*pn)(2) - (*totalStation)(2);
	distance = sqrt(xia*xia + yia*yia + zia*zia);
	return distance;
}

Vector4d totalStationMeasuredPosition(Vector3d * totalStation, Vector4d * pn, Vector3d
* tsr,
	default_random_engine * g, normal_distribution<double> * dd,
	normal_distribution<double> * da, normal_distribution<double> * db)
{
	double alpha = totalStationPrismToHrztAngle(totalStation, pn) + (*da)(*g);
	double beta = totalStationPrismToVtclAngle(totalStation, pn) + (*db)(*g);
	double distance = totalStationPrismToDistance(totalStation, pn) + (*dd)(*g);

	double xisa = distance * cos(beta) * sin(alpha);
	double yisa = distance * cos(beta) * cos(alpha);
	double zisa = distance * sin(beta);

	double xis = xisa + (*totalStation)(0) + (*tsr)(0);
	double yis = yisa + (*totalStation)(1) + (*tsr)(1);
	double zis = zisa + (*totalStation)(2) + (*tsr)(2);

	Vector4d measuredPosition(xis, yis, zis, 1.0);

	return measuredPosition;
}
```

```
void statisticLoop(stringstream * ssE, stringstream * ssN, stringstream * ssU, Vector4d * idealInvert, default_random_engine * g, normal_distribution<double> * dtsr, normal_distribution<double> * dd,
	normal_distribution<double> * da, normal_distribution<double> * db)
{
	// Inner diameter of the tunnel
	double ID = 2.34;
	// Total length of the TBM
	double TBMLength = 4.652;
	// True coordinates of the total station
	Vector3d totalStation(27740.564, 5934907.978, 643.248);

	// True coordinates of prisms in the b-frame
	Vector4d p1b; p1b << 0.408, TBMLength/2.0 - 4.580, 0.814, 1.0;
	Vector4d p2b; p2b << 0.695, TBMLength/2.0 - 4.579, 0.751, 1.0;
	Vector4d p3b; p3b << 0.797, TBMLength/2.0 - 4.581, 0.476, 1.0;

	// True coordinates of prisms in the n-frame
	Vector4d p1n; p1n << 27740.618, 5934807.717, 644.163, 1.0;
	Vector4d p2n; p2n << 27740.332, 5934807.712, 644.099, 1.0;
	Vector4d p3n; p3n << 27740.231, 5934807.712, 643.823, 1.0;

	// Registration error of total station
	Vector3d tsr((*dtsr)(*g), (*dtsr)(*g), (*dtsr)(*g));

	//cout << "Total Station Registration Error" << endl << tsr << endl << endl;

	// Simulate the measurement process of a total station
	Vector4d p1nm = totalStationMeasuredPosition(&totalStation, &p1n, &tsr, g, dd, da, db);
```

```
	Vector4d p2nm = totalStationMeasuredPosition(&totalStation, &p2n, &tsr, g, dd,
da, db);

	Vector4d p3nm = totalStationMeasuredPosition(&totalStation, &p3n, &tsr, g, dd,
da, db);


	Matrix4d Fb3p = threePointstoFrame (&p1b, &p2b, &p3b);

	Matrix4d Fn3p = threePointstoFrame (&p1nm, &p2nm, &p3nm);


	Matrix4d Fnb = Fn3p * Fb3p.inverse();

	Matrix4d Fnb_wr = bodyFrameToBodyFramewithoutRolling(&Fnb);


	Vector4d i1b;

	double x1 = cos(3.0/2.0 * PI) * ID/2;

	double y1 = -TBMLength/2.0;

	double z1 = sin(3.0/2.0 * PI) * ID/2;

	i1b << x1, y1, z1, 1.0;


	Vector4d i1n = Fnb_wr * i1b;

	*ssE << (i1n.x() - idealInvert->x()) * 1000 << endl;

	*ssN << (i1n.y() - idealInvert->y()) * 1000 << endl;

	*ssU << (i1n.z() - idealInvert->z()) * 1000 << endl;

}


Vector4d getIdealInvert()

{

	// Inner Diameter of the tunnel

	double ID = 2.34;

	// Total length of the TBM

	double TBMLength = 4.652;
```

```
        // True coordinates of prisms in the b-frame
        Vector4d p1b; p1b << 0.408, TBMLength/2.0 - 4.580, 0.814, 1.0;
        Vector4d p2b; p2b << 0.695, TBMLength/2.0 - 4.579, 0.751, 1.0;
        Vector4d p3b; p3b << 0.797, TBMLength/2.0 - 4.581, 0.476, 1.0;

        // True coordinates of prisms in the n-frame
        Vector4d p1n; p1n << 27740.618, 5934807.717, 644.163, 1.0;
        Vector4d p2n; p2n << 27740.332, 5934807.712, 644.099, 1.0;
        Vector4d p3n; p3n << 27740.231, 5934807.712, 643.823, 1.0;

        Matrix4d Fb3p = threePointstoFrame (&p1b, &p2b, &p3b);
        Matrix4d Fn3p = threePointstoFrame (&p1n, &p2n, &p3n);

        Matrix4d Fnb = Fn3p * Fb3p.inverse();
        Matrix4d Fnb_wr = bodyFrameToBodyFramewithoutRolling(&Fnb);

        Vector4d i1b;
        double x1 = cos(3.0/2.0 * PI) * ID/2;
        double y1 = -TBMLength/2.0;
        double z1 = sin(3.0/2.0 * PI) * ID/2;
        i1b << x1, y1, z1, 1.0;

        Vector4d i1n = Fnb_wr * i1b;
        cout << i1n << endl;
        return i1n;
}
```

```
int main()
{
	Vector4d idealInvert = getIdealInvert();

	const int nrolls = 500;
	default_random_engine g;
	normal_distribution<double> dtsr(0.0, 0.005);
	normal_distribution<double> ddistance(0.0, 0.00115);
	normal_distribution<double> dalpha(0.0, PI/SecondsPerPi);
	normal_distribution<double> dbeta(0.0, PI/SecondsPerPi);

	stringstream ssE; ssE.precision(15);
	stringstream ssN; ssN.precision(15);
	stringstream ssU; ssU.precision(15);

	for(int i = 0; i < nrolls; i++)
	{
		statisticLoop(&ssE, &ssN, &ssU, &idealInvert, &g, &dtsr, &ddistance,
&dalpha, &dbeta);
	}

	ofstream outputFile ("output.txt");
	if (outputFile.is_open())
	{
		outputFile.precision(15);
		outputFile << "Easting" << endl << ssE.str() << endl << endl;
		outputFile << "Northing" << endl << ssN.str() << endl << endl;
		outputFile << "Up" << endl << ssU.str() << endl << endl;
		outputFile.close();
```

```
        }
    }
```