# NOTE TO USERS

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Tomasz Byczkowski
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

A Stereo-Based System with Inertial Navigation for 3D Scanning of Outdoor Objects and Architecture
TITRE DE LA THÈSE / TITLE OF THESIS

J. Lang
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

P. Payeur

A. Whitehead

Dr. S. Cooke

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

# A Stereo-Based System with Inertial Navigation for 3D Scanning of Outdoor Objects and Architecture

by

Tomasz Byczkowski

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

*Your file Votre référence*
*ISBN: 978-0-494-51638-6*
*Our file Notre référence*
*ISBN: 978-0-494-51638-6*

# Abstract

Three-dimensional scanning is a topic which, in recent years, has gained significant attention from several research fields, including industrial design, reverse engineering and prototyping (*e.g.*, CAD), body imaging (*e.g.*, fashion and health care), computer vision (*e.g.*, world mapping), entertainment (*e.g.*, gaming, visual effects and animation), photo tourism, forensics and documentation of cultural artifacts. Currently, 3D scanning is an involved process that generally requires special hardware and procedures, and is therefore frequently limited to a laboratory environment. However, there is a world full of potential subjects to digitize, most of which cannot be relocated to a laboratory without considerable expense, or at all.

In this thesis, we introduce a 3D scanning system suitable for scanning objects and architecture using a mobile stereoscopic camera combined with an inertial navigation sensor. The inertial navigation system acts as a coarse estimator for rapid registration and camera localization. Our scanning apparatus is assembled entirely using commercially available hardware, and the scanning process itself requires no technical skill and our system may be adapted to a wide variety of scanning conditions. We also introduce a set of heuristics for multi-view data registration that aim to produce reliable and repeatable convergence despite the diversity of potential subjects, and different types of motions that may be executed by the operator during the scanning process. A surface reconstruction technique is used to generate a mesh, and subsequent texturing of the model is done through interpolation of the per-vertex colour information extracted from the rectified colour images recorded by the camera. By texturing the surface of an object in a 3D application, the realism of the product is greatly increased. The results presented will show that our system is a viable solution for reproducing fully textured polygonal models with low surface deviation and realistic visual appearance.

## Acknowledgements

I would like to thank my thesis supervisor Dr. Jochen Lang, whose expertise, patience and resources made this thesis possible.

I also acknowledge the financial support from Ontario Graduate Scholarship Program (OGS).

# Contents

# List of Tables

# List of Figures

# Glossary of Terms

| | |
|---|---|
| **2D** | Two-dimensional. |
| **3D** | Three-dimensional. |
| **AD** | Absolute intensity differences. |
| **AHRS** | Attitude and heading reference system. |
| **API** | Application programming interface. |
| **BP** | Belief propagation. |
| **CAD** | Computer aided design. |
| **CCD** | Charge-coupled device. |
| **DOF** | Degrees of freedom. |
| **DLL** | Dynamic link library. |
| **DP** | Dynamic programming. |
| **FPS** | Frames per second. |
| **GC** | Graph cut. |
| **GPS** | Global positioning system. |
| **HFOV** | Horizontal field of view. |
| **HLA** | High level architecture. |
| **ICP** | Iterative closest point algorithm. |
| **INS** | Inertial navigation system. |
| **iSM** | Instant scene modeler. |
| **LSQ** | Least squares. |
| **ML** | Maximum likelihood. |
| **NURBS** | Non uniform rational B splines. |
| **PGR** | Point grey research. |
| **PLY** | Stanford Polygon file format. |
| **RANSAC** | Random sample consensus. |
| **RMS** | Root mean squared. |
| **RP** | Rapid prototyping. |

| | |
|---|---|
| **SA** | Simulated annealing. |
| **SAD** | Sum of absolute intensity differences. |
| **SD** | Squared intensity differences. |
| **SDK** | Standard development kit. |
| **SIFT** | Scale invariant feature transform. |
| **SLAM** | Simultaneous localization and mapping. |
| **SO** | Scanline optimization. |
| **SVD** | Singular value decomposition. |
| **TAD** | Sum of truncated absolute intensity differences. |
| **USB** | Universal serial bus. |
| **WTA** | Winner–take–all. |

# Chapter 1

# Introduction

## 1.1 3D Scanning

A 3D scanner is a device that analyzes an object or environment in the real world to collect data on its shape and appearance in order to reconstruct a digital replica. Currently, three-dimensional scanning is an involved process that often requires specialty hardware and methods, and is therefore frequently limited to controlled indoor environments.

Three-dimensional scanning techniques are generally based on different principles and capture a variety of complementary information. The highest level categories for these techniques are contact and non-contact methods, and the non-contact methods can be further sub-categorized into active and passive methods [57]. Contact scanners probe the subject using physical touch, while non-contact scanners measure the light reflected by the subject or environment. A non-contact technique is classified as active "when the measured object must be actively illuminated, while passive techniques work with ambient light [57]." Our interest lies in non-contact techniques because optical scanners offer a more powerful alternative to the conventional data collection methods because they are more resolute and offer more accurate distance measurements.

The most common active methods used for 3D measuring employ a laser light source for optical triangulation, or pulsed or modulated time-of-flight, or interferometric methods such as holography. Structured light is another active scanning method that functions with alternative light sources [57]. Triangulation uses the geometric relation between the direction of the emitted laser beam and the direction of the detected reflection to determine the position of the object surface points. In actuality, it is a laser line on the object, not a beam, that is swept along the object surface [58]. The time of flight method uses

the known value of the speed of light in air, and detects the distance to a point on the object surface by measuring the time interval between the emission of a laser pulse and the detection of its reflection. In this case, the scanning system sweeps the object surface in several parallel lines while the laser head emits pulses at a high rate, producing a matrix of points on the object surface [58]. Conoscopic holography is an implementation of a polarized light interference process based on crystal optics. In the basic interference setup, a light beam is projected onto a diffusive object where it creates a light point on a target and disperses the light in all directions. A complete solid angle of the diffused light is analyzed and the measurement process corresponds to the retrieval of a distance of the light point from a fixed reference plane [78].

Conversely, common passive techniques include stereo vision and structure from motion [57]. Structure from motion is a technique that requires only a single calibrated camera to compute a point in space given its image in two views and the camera matrices of those views. The solution requires the definition and minimization of some cost function to estimate a solution to a multi-view triangulation that is invariant to projective transformations in space [32].

We wish to propose an additional non-contact passive method for stereo range finding that includes inertial navigation to aid in registration during the reconstruction of a 3D model. Its primary purpose will be to capture and model outdoor targets such as statues and architecture, which requires the system to be portable enough to be a wearable system. Our proposed system aims to use only a commonly available commercial stereoscopic charge-coupled device (CCD) camera to collect range and colour data about the subject. The primary disadvantage of active scanners that make them unsuitable for outdoor applications is that they are often elaborate, awkward, or require a strong power source. Also, laser scanners generally require a turntable or other rigid motion for registration.

Widespread applications for this technology include industrial design, reverse engineering and prototyping (*e.g.*, CAD), body imaging (*e.g.*, fashion and health care), computer vision (*e.g.*, world mapping), entertainment (*e.g.*, gaming, special effects and animation), photo tourism, forensics and documentation of cultural artifacts.

## 1.2 Common Applications

According to Willis, Speicher and Cooper [95], it has "become increasingly important to be able to generate free-form 3D shapes in commercial applications using rapid pro-

totyping technologies" because in many cases the shapes of interest are taken from real objects that do not have pre-existing computer models, and an accurate model would be too expensive and time consuming to create by hand. Subsequently, companies have begun using 3D scanners to measure exemplars of their desired shape. Rapid prototyping machines (RP) are employed to generate test exemplars of the modeled object prior to mass production. The reasoning behind the use of 3D scanning in reverse engineering and industrial design applications is basically the same.

Treleaven and Wells [85] claim that "3D body-surface scanners are transforming our ability to accurately measure and visualize a persons body size, shape and skin-surface area." Since the surface area of the body is strongly related to certain physiological processes (*e.g.*, dialysis rates and water turnover), 3D whole-body scanning seems to offer great potential for health care applications in epidemiology (*e.g.*, anthropometric surveys), diagnostics (*e.g.*, deformity detection and skin analysis), treatment (*e.g.*, burns, orthotics and prosthetics), and monitoring (*e.g.*, exercise and diet). Body–surface area is often a factor when a medical professional prescribes certain drugs, dialysis rates, and other similar treatments because body size and shape may imply different states of health or disease.

Paquette [55] refers to "apparel on demand" or "mass-produced custom" as the process that seeks to produce clothing designed and fitted to an individuals size and proportions. Body scanning would produce better fitting garments and reduce the cost of manufacturing labour and the cost of overhead for infrequently stocked sizes.

Pezatti and Fontana [57] explain that optical techniques play an important role in the field of cultural heritage diagnostics, because of their safety and effectiveness. A contact diagnostic process is particularly unfavourable when documenting aged artifacts because the act of probing its surface could potentially damage it in the process. This is why optical techniques are presently used to measure buildings, statues, inscriptions, coins, and even paintings. The uses for high density digital models range from historical and artistic studies to the assembly of easy access and long–lasting digital archives for use in conservation, and restoration projects. Computer aided restoration facilitates the visualization of any change resulting from a proposed restoration treatment, including any proposed reconstructions of missing or damaged elements. Furthermore, by measuring the shape over time it is possible to obtain data on the object alterations [27]. Such analysis can reveal mechanical stresses, quantify the effects of microclimatic variations (*e.g.*, temperature and/or humidity), and measure surface degradation or any shape variations introduced by a restoration project. Moreover, the digital model constructed with

3D data can be used for virtual reality applications, and the work of art can be enjoyed outside its usual surroundings.

Se & Jasiobedzki [73] foresee a use for 3D scanning in crime scene reconstruction because it is normally a very labour intensive process that results in a lot of time spent at a crime scene taking many measurements and photographs. Not only that, but additional time is spent inputting all of the data into a computer manually. They believe that automatic generation of photo-realistic 3D models using a hand-held device would be highly desirable to crime scene investigators.

Lastly, the application of 3D scanning in the fields of video game development, animation and special effects helps to produce a more detailed and realistic experience for the audience while expediting content creation [56]. Digital models of real people and objects can be produced and inserted into simulated environments that are too dangerous, too fantastic, or too costly to create in the real world.

## 1.3 Inertial Navigation

Traditional inertial systems have historically been heavy, bulky and costly. Recent advances in microelectromechanical (MEMS) technology have made accurate, compact, low-power inertial navigation systems (INS) commercially available at reasonable cost [1, 2]. The primary function of an inertial tracking device in a 3D model acquisition scenario is to provide a gyro-enhanced attitude and heading reference for the probe that actually performs the capture, which in our case is a stereoscopic camera. The INS provides a drift-free measure of the orientation of the stereo camera in the real world that facilitates quicker and more accurate registration during model reconstruction. Also, integrating the discrete acceleration data permits localization of the probe in a common world frame following hand-eye calibration as required when a sensor is mounted on an end effector.

According to Kelly [39], these devices exhibit a large number of characteristics that ease their widespread use. They are a preferred navigation aid in military applications because they radiate no energy that may assist an adversary in locating the sensor. Also, they require no information from the environment, so they are indifferent to weather conditions, and the attendant poor visibility of landmarks, and they cannot be jammed or interfered with. They can even be used on vehicles travelling at high enough speeds to ionize the surrounding air, and effectively jam their own navigation radar. They can be used easily in unsurveyed areas where landmark positions are unknown, in featureless

terrain where there are no landmarks, and in situations where artificial landmarks cannot be installed for any of a variety of reasons. They are ubiquitous components in missiles, tanks, satellites and long haul commercial aircraft. Soon inertial navigation systems may even become robust enough and cheap enough to become standard equipment on automobiles in much the same way as its cousin, the global positioning system (GPS).

Inertial navigation systems are important to mobile robotics because "they provide the most accurate form of dead reckoning currently available [39]." Also, the use of inertial principles implies that these devices can be used literally anywhere in the universe, including the entire surface of the globe, and the atmosphere, provided the local gravitational field is known. They are feasible alternatives in many places where other types of systems cannot be used, including interplanetary space, underground, and undersea. In this respect, they are the most generally applicable navigation technology in existence [39].

## 1.4 Thesis Statement

Good quality mobile scanning can be realized using the standard 3D scanning pipeline in conjunction with a stereo camera. The use of motion tracking instrumentation and registration techniques is sufficient enough so that no extra visual tracking is required to produce a model from multi-view data, if care is taken.

### 1.4.1 Major Contributions

We have developed a mobile 3D scanning system capable of scanning objects and architecture in any setting. One advantage of our system is that we do not restrict the scanning process with the need for elaborate capture rigs or rigid motion, therefore our system is mobile and does not face restrictions related to the workplace. Our system is implemented entirely using commercially available hardware and the scanning process itself requires little technical skill on part of the operator and our system may be adapted to fit a wide range of scanning conditions. Data from multiple views are ultimately fused into a single static surface model by a set of heuristics that seek to maintain functionality amid varying types of subjects that may be encountered, as well as different types of motions that may be executed by the operator during the scanning process.

In summary, the key contributions arising from our work are:

- a mobile, cost-effective system for modeling outdoor objects and architecture, and

- development of a heuristic algorithm for multi-view registration of surface point clouds which aims to maintain generality when scanning different types of subjects.

## 1.5 System Overview

Our system aims to be able to produce textured 3D models from video of outdoor targets: specifically statues and architecture. This goal requires that the system be portable enough in terms of size, weight and power requirements to implement as a wearable system. This also implies that data capture should not require a turntable or other rigid motion like the type often encountered using a 3D laser scanner. The inertial navigation system acts as a coarse registration estimator for faster fine registration and camera localization. The entire system requires minimal processing power for post-processing, sensor fusion and data registration.

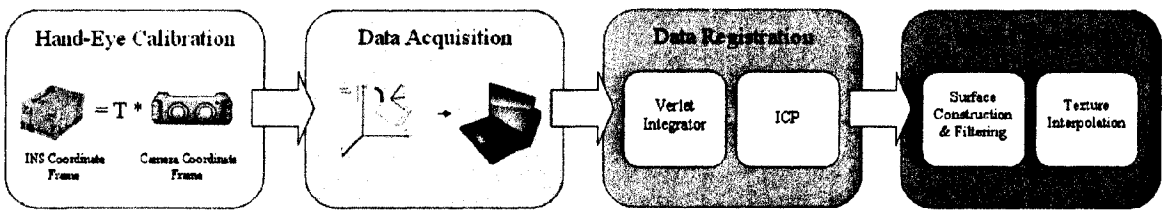



Figure 1.1: Flow-chart or the high level architecture of our scanning system.

Figure 1.1 shows the high level architecture (HLA) for the system in four stages. These stages are categorized as: hand-eye calibration, data acquisition, data registration, and model synthesis. The purpose of the first stage is to determine the spatial relationship between the camera and navigation sensor when mounted to the same rigid body. This is achieved by acquiring a sequence of disparity images for a specific calibration target whose measurements are known. The transformation matrix between the camera and the INS coordinate frames is then sent down the scanning pipeline to the next stage.

The data acquisition phase simply records rectified colour images, stereo information (*e.g.,* disparity images), accelerometer readings, and camera orientation. The camera and the INS sensor are mounted together in some rigid fashion, and a laptop equipped with IEEE 1394a and USB connectivity is used as the host and storage. All of this data is sent down the pipeline to later stages in the system.

The data registration stage is where all of the sensor readings are fused to construct a static model of the subject. The accelerometer data is filtered to reduce noise and

drift, then integrated to produce position change estimates. The orientation data is assumed to be drift–free, according to manufacturer specifications. Three-dimensional vertex information is extracted from the disparity data using stereo vision techniques, and the orientation data is used to transform all of the views into a single common coordinate frame. Once in a common coordinate frame, the iterative closest point (ICP) algorithm is used to register all of the individual point clouds into their correct alignment relative to each other. The resultant vertices and their corresponding per–vertex colour information are stored using the PLY (*i.e.*, Stanford Polygon) file format prior to model synthesis.

The model synthesis stage consists of surface reconstruction, texture interpolation, filtering and cleaning of the final model. Surface reconstruction refers to generating a polygonal surface for the model from the vertex information received from the previous stage, and the method used by our system is the ball-pivot algorithm. Texturing of the model is done through interpolation of the per-vertex colour data extracted from the rectified colour images recorded during data capture. Additional filtering and cleaning of the model simply improves the visual quality of the end result and involves removing faces and vertices that are superfluous or appear incorrect. The stages and their components are explained in greater detail in the following chapters.

Related work in the areas of stereo vision and multi-view registration of measurement data is given in Chapter 2. It also describes work related to hand-eye calibration methods, and local and global tracking technologies. Chapter 3 details the hand-eye calibration technique used by our system. Chapter 4 describes the steps taken in data registration process employed by our system. That includes the filtering of raw data, the heuristics used for scene matching, and texture interpolation from video. Results and concluding remarks are presented in Chapter 5 and Chapter 6, respectively.

# Chapter 2

# Related Work

In this chapter, we discuss previous work in the field of 3D scanning and model reconstruction. Section 2.1 introduces some of the fundamentals of stereo vision, and the techniques used to extract the depth information required to create a virtual reconstruction of a subject or environment. Sensor calibration methodology is presented in Section 2.2. The discussion focuses mainly on the use of quaternions to determine the relationship between a sensor and an end effector, but mentions classical efforts that do not use quaternions. Section 2.3 gives an in-depth discussion of the iterative closest point algorithm used for aligning adjacent views captured by the camera, as well as some of the advantages and disadvantages of its various implementations. Section 2.4 reviews some of the other mobile 3D scanning systems with a focus on those that use real-time positional tracking methods to localize a second sensor in real space, but not exclusively so. Finally, we summarize this chapter in Section 2.5.

## 2.1 Stereo Vision Fundamentals

Stereo vision is a common method used for range finding in computer vision, and it uses one of the same basic principles that human vision (*i.e.*, binocular vision) does to perceive depth. Essentially, two cameras take pictures of the same scene, but they are separated by a distance – just like human eyes. The result is two slightly different images of the same scene. The images are then compared by shifting the two images over top of each other to find any matches between them. The shifted amount is called the *disparity*: a term first introduced in human vision literature to describe the difference in location of corresponding features seen by the left and right eyes [46, 71]. The disparity at which

objects in the images best match is used to calculate their distance from the camera. We will discuss some of the popular stereo matching algorithms available using the taxonomy and categorization scheme provided by Scharstein & Szeliski [71] as a level playing field.

In order to support an informed comparison of stereo matching algorithms, Scharstein & Szeliski developed a taxonomy to compare them within a common framework. Their framework focuses on *two-frame* stereo correspondence algorithms that produce a dense univalued disparity function $d(x, y)$ as output (*i.e.*, a value at each pixel), because this type composes the majority of previous work. They presented a set of "building blocks" from which a large set of existing algorithms could be constructed. These were based on the observation that stereo algorithms generally perform some, or all of the following four tasks [69, 70]:

1. Matching cost computation,
2. Cost (support) aggregation,
3. Disparity computation/optimization, and
4. Disparity refinement.

As a supplement, they developed a stand-alone, modular C++ evaluation of multiple stereo algorithms that is closely tied to the four parts of the taxonomy. The default implementation included support for several matching cost computations, such as squared intensity differences (SD) [3, 31, 77], absolute intensity differences (AD) [36], truncated quadratics and contaminated Gaussians [9, 10, 70], gradient-based measures [68, 75], normalized cross-correlation [31, 67, 12], and a variety of binary matching costs [47, 4, 52]. Cost aggregations initially supported by the implementation included several window-based algorithms [38, 11, 14], and diffusion algorithms [82, 76, 70]. The disparity computation/optimization options available included a winner-take-all (WTA) approach (choose the lowest matching cost) [71], dynamic programming [71, 11], scanline optimization [71], simulated annealing [28, 48, 5], or graph cuts [15]. Additional examples of each can be found in [71], and its associated references. The disparity refinement step was optionally excluded from the evaluations because it often represents some type of post-processing, and they prefer to measure the performance of the raw algorithm components.

However, the software can also be easily extended to include any newly developed algorithms or their components. Once integrated, the performance of an algorithm can

easily be compared with existing ones by evaluating the resultant disparity error and reprojection error [71]. This work is the basis for the highly successful Middlebury stereo test bed [72], which provides on-line rankings of over fourty of the most popular algorithms. This makes it an invaluable tool when shopping for a stereo correspondence solution for our application. Breaking down the work flow of any given algorithm allows us to characterize its components and how they fit into the aforementioned taxonomy.

According to Middlebury, one of the best ranked algorithms was proposed by Klaus et al. [41]. First, it utilizes colour segmentation on the reference image prior to computing its matching cost measure. At its core, the matching cost measure is a unique combination of sum of absolute intensity differences (SAD) and a gradient based measure, followed by cost aggregation which is performed over a $3 \times 3$ window, and a back-and-forth validation [26] procedure to ensure that the left-to-right and right-to-left disparity maps agree. One of the contributions of this work is a novel extension to the evaluation test bed, whereby the disparity computation does not only assign a disparity value to each pixel, in the traditional sense. Rather, a disparity plane is assigned to each colour segment by applying a novel robust plane fitting method applied using loopy belief propagation (BP) [24] optimization.

Another well-ranked algorithm evaluated using the test bed was proposed by Yang et al. [98]. Unlike the previous algorithm, this one makes far greater use of the extensibility of the evaluation test bed. It begins by computing a colour-based weighting for each pixel that serves in producing colour regions, as well as in the cost matching measure. The matching cost measure used in this algorithm is also basically an absolute intensity difference measure, but it is modified by the colour-based weightings in the RGB colour space. The task of cost aggregation is performed using an adaptive window, the benefit of which is improved results in both smooth and discontinuous regions. A back-and-forth validation step is also included in this algorithm. Lastly, the disparity computation, in this case, is performed using a novel plane-fitting procedure that employs hierarchical belief propagation (BP) [94].

One of the more recent algorithms that actually did not rank very well on this test bed was proposed by Mattoccia et al. [84]. Their ranking is likely a result of the fact that their approach is aimed at maximizing the speed-accuracy trade-off rather than focusing entirely on the quality of the results. While this algorithm employs local colour information to improve its matching cost measure, it does not include a global segmentation step. The cost measure used is the sum of truncated absolute intensity differences (TAD), with an adaptive window cost aggregation, and it does perform back-and-forth

cross-checking for disparity map validation. Additional speed gains occur in the disparity optimization step where plane-fitting is reserved for untextured (or poorly textured) segments. The authors note that either scanline optimization or belief propagation could be used effectively with the plane-fitting in this algorithm.

An earlier method proposed by Stefano & Mattoccia [79] that is not evaluated on the Middlebury page was designed to be the VIDET (VIsual DEcoder by Touch) [13] project's PC-based, real-time, stereo vision system. In this system, normalized images are matched based on the SAD measure; each point of the left image is associated with a point of the right image by searching for the minimum SAD within a given disparity range ($d_r = d_{max} - d_{min}$). The cost aggregation is performed by a standard square window like many popular methods, and also includes a back-and-forth validation step so as to avoid to assign wrong disparities to occluded points. The disparity computation step is performed using a sub-pixel validation with a resolution of $\frac{1}{8}$-pixel which scales the SAD disparity values from the original range $[d_{max}, d_{min}]$ to the range $[8d_{max}, 8d_{min}]$. The work done by Stefano & Mattoccia is very similar to the stereo correlation used by our system, and actually aims to improve upon the real-time performance of PC-based systems like the one offered by Point Grey Research (PGR) [59].

| Method | Matching Cost | Aggregation | Optimization | Uses Colour Information | Back-and-Forth Validation |
|---|---|---|---|---|---|
| Klaus et al. [41] | absolute differences | square window | loopy belief propagation | ✓ | ✓ |
| Yang et al. [98] | absolute differences | adaptive window | hierarchical belief propagation | ✓ | ✓ |
| Mattoccia et al. [84] | truncated absolute differences | adaptive window | scanline optimization, or belief propagation | ✓ | ✓ |
| Stefano & Mattoccia [79] | absolute differences | square window | sub-pixel validation | – | ✓ |
| PGR Triclops | absolute differences | square window | sub-pixel, and surface validation | – | ✓ |

Table 2.1: Summary of the components of each stereo correlation algorithm discussed in Section 2.1 and classified according to the taxonomy of Scharstein & Szeliski.

The taxonomies of these algorithms are summarized in Table 2.1. Based on this information and the performance rankings of the algorithms shown in 2.2, it is apparent that an absolute intensity differences matching cost measure with either a fixed or adaptive window is a fast, simple and reliable method for computing dense disparity maps.

Our system utilizes the PGR Triclops API, which includes these processes as well, so its performance should be comparable to these other approaches. In many cases, it appears that the disparity computation/optimization step makes or breaks the performance of the algorithm overall. As with other stereo algorithms, the quality (accuracy, coverage and number of outliers) of the depth data depends on the presence of texture in the images. In summary, it appears highlighted in the same table.

Table 2.2 shows Middlebury evaluation data for ten of the most recent stereo correspondence algorithms ordered by rank (lower is better). They represent a subset of available algorithms, and were selected to illustrate a wide range of results including a few representing the current state of the art. Comparing these results against the taxonomies of the same algorithms indicates some correlation between the techniques it employs and its effectiveness. Those discussed earlier in this section are highlighted. The columns are sorted by each of the four data sets used in the evaluation: Tsukuba, Venus, Teddy and Cones [72]. The ground truth disparity map of each data set is also divided into three subsets: non-occluded regions (denoted "nonocc"), regions near depth discontinuities or near occluded regions (denoted "disc"), and non-occluded or half-occluded pixels (denoted "all"). The values in each column represent the average percentage of bad pixels produced by an algorithm in each of these regions, for each data set.

| Algorithm | Avg. Rank | Tsukuba | | | Venus | | | Teddy | | | Cones | | | Average % of Bad Pixels |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | nonocc | all | disc | nonocc | all | disc | nonocc | all | disc | nonocc | all | disc | |
| Klaus et al. [41] | 3.1 | 1.11 | 1.37 | 5.79 | 0.10 | 0.21 | 1.44 | 4.22 | 7.06 | 11.8 | 2.48 | 7.92 | 7.32 | 4.23 |
| Wang & Zheng [93] | 3.1 | 0.87 | 1.16 | 4.61 | 0.11 | 0.21 | 1.54 | 5.16 | 8.31 | 13.0 | 2.79 | 7.18 | 8.01 | 4.41 |
| Yang et al. [98] | 4.1 | 0.88 | 1.29 | 4.76 | 0.13 | 0.45 | 1.87 | 3.53 | 8.30 | 9.63 | 2.90 | 8.78 | 7.79 | 4.19 |
| Xu & Jia [97] | 4.8 | 0.88 | 1.43 | 4.74 | 0.18 | 0.26 | 2.40 | 5.01 | 9.12 | 12.8 | 2.78 | 8.57 | 6.99 | 4.60 |
| Taguchi et al. [83] | 11.0 | 1.69 | 2.04 | 5.64 | 0.14 | 0.20 | 1.47 | 7.04 | 11.1 | 16.4 | 3.60 | 8.96 | 8.84 | 5.66 |
| Mattoccia et al. [49] | 14.3 | 1.29 | 1.71 | 6.83 | 0.25 | 0.53 | 2.26 | 7.02 | 12.2 | 16.3 | 3.90 | 9.85 | 10.2 | 6.09 |
| Mattoccia et al. [84] | 27.5 | 1.16 | 2.11 | 6.06 | 4.03 | 4.75 | 6.43 | 9.04 | 15.2 | 20.2 | 5.37 | 12.6 | 11.9 | 8.24 |
| Mordahai & Medioni [51] | 32.0 | 3.79 | 4.79 | 8.86 | 1.23 | 1.88 | 11.5 | 9.76 | 17.0 | 24.0 | 4.38 | 11.4 | 12.2 | 9.25 |
| El-Etriby et al. [22] | 43.6 | 4.26 | 6.53 | 15.4 | 6.71 | 8.16 | 26.4 | 14.5 | 23.1 | 25.5 | 10.8 | 20.5 | 21.2 | 15.3 |
| Olague et al. [54] | 47.0 | 7.95 | 9.54 | 28.9 | 4.41 | 5.53 | 31.7 | 17.7 | 25.1 | 44.4 | 14.3 | 21.3 | 38.0 | 20.7 |

Table 2.2: Middlebury evaluation test bed results for several different algorithms. These represent a small subset of available algorithms, and were selected to illustrate a full range of results including the state of the art. Comparing these results with information from Table 2.1 indicates some correlation between the techniques employed by an algorithm and its overall effectiveness. Table adapted from [72].

## 2.2 Sensor Calibration

Whenever a camera is mounted to a rigid platform or end effector, it is important to know the relationship between the camera and the platform or end effector, and the determination of that relationship is known as the hand–eye calibration problem. More specifically, it is the task of computing the relative homogeneous transformation between two coordinate frames, one centered at the camera lens center, and the other at the end effector [88]. The relationship is important in our system because it will be used to map camera centered measurements into a common workspace frame with the help of an inertial navigation system.

An early solution to this problem was proposed by Tsai & Lenz [88] for use with robotic arms. Their work was motivated by the fact that previous methods using global non–linear optimization was too time consuming because of the large number of unknowns associated with robot kinematic models, and it was not guaranteed to converge. Furthermore, redundant images could not be used to reduce error because the computation would become too prohibitive.

Under the proposed formulation, the number of unknowns stays the same no matter how many frames are used simultaneously. The idea was a simple one: a robot carrying a camera makes a series of motions with the camera and taking a picture of a known object at the pause of each motion. The position, geometry and visual characteristics of this calibration object are known very accurately relative to an arbitrarily selected coordinate system setup on the object [43, 86, 87]. It attempts to solve a series of homogeneous matrix equation of the form:

$$AX = XB \tag{2.1}$$

Here, $A$ represents a transformation from the calibration frame to the camera frame, $B$ represents the transformation from the hand frame to the world frame, and $X$ represents the transformation from the hand frame to the camera frame, as illustrated in Figure 2.1. They are each $4 \times 4$ homogeneous transformation matrices of the following form:

$$A = \begin{pmatrix} R_A & t_A \\ 0 & 1 \end{pmatrix}$$

In the robotics application for which their algorithm was originally intended, the transformation $B$ can be further decomposed into a set of $[B_1 \ldots B_i]$ representing the kinematic positions of each rigid section of a robot arm. However, in the general case, if $A_1$ and $A_2$ are the transformations from the object frame to the hand frame in two

different positions, then:

$$A = A_2 A_1^{-1} \tag{2.2}$$

Also, if $B_1$ and $B_2$ are the transformations from the hand frame to the base frame in two different positions, then:

$$B = B_2^{-1} B_1 \tag{2.3}$$

The solution for this classical approach begins with two equations:

$$R_A R_X = R_X R_B \tag{2.4}$$

and

$$(R_A - I)t_X = R_X t_B - t_A \tag{2.5}$$

In order to solve these equations, it is necessary to have at least three views. Algebraic and geometric properties of rotation (orthogonal) matrices are used to solve these equations. If $n_A$ and $n_B$ are eigenvectors associated with eigenvalue 1 of $R_A$ and $R_B$ respectively, then Equation 2.4 can be written as:

$$\begin{aligned} R_A &= R_X R_B R_X^T \\ R_A R_X n_B &= R_X R_B n_B \\ &= R_X n_B \end{aligned} \tag{2.6}$$

$$n_A = R_X n_B \tag{2.7}$$

One of the things Tsai & Lenz suggested was to represent $R_X$ by its unit eigenvector $n_X$ and an angle $\theta_X$. They noticed the following two properties:

$$n_X \cdot (n_A - n_B) = 0$$

and

$$(n_A - n_B) \cdot (n_A + n_B) = 0$$

This leads to the following alternate expression of Equation 2.7:

$$(n_A + n_B) \times n = n_A - n_B \tag{2.8}$$

$$n = \left( \tan \frac{\theta_X}{2} \right) n_X$$

To conclude, for the general case of $n$ motions, this algorithm allows one to solve for an over constrained set of $2n$ linear equations in three unknowns to determine the final calibration information. While this work was done with a robotic arm in mind, it is

fully transferable to our INS assisted mobile scanning system. However, in the presence of noise, the solution available with the linear system is not good. Errors may be due to camera calibration inaccuracies or inexact knowledge of the relationship between the base frame and the hand frame [33].

In order to simplify the solution to the classical hand–eye calibration problem, Chou & Kamel [18] proposed that the equation $AX = XB$ should be transformed into its equivalent form in terms of quaternions: a notation first used by Hamilton [30]. Quaternion algebra provides a simpler, unique, and elegant representation for describing finite rotations in space than ordinary vector algebra. Chou & Kamel argue that this transformation changes it into a simple and well-structured linear system of equations whose general solution can be determined with (but does not necessarily require) an inverse method incorporating singular value decomposition (SVD) [80]. Of particular interest in this work is the inference that 3-vectors (such as eigenvectors) can be represented as purely imaginary quaternions, or "vector quaternions". Composing quaternion rotations is also trivial and computationally cheaper, and accurate than using orthonormal matrices, and makes it possible to find a closed-form solution to the problem.

Daniilidis [21] proposed a similar method using dual quaternion parametrization, which is informally known as the "screw method". The advantages of that approach include the simultaneous computation of rotation and translation, and its ability to solve singular configurations naturally where conventional approaches fail (*e.g.*, calibration of cameras mounted on vehicles).

Unlike other work done up to that point that focused almost exclusively on the classical formulation, Horaud & Dornaika [33] reformulated the solution for the hand-eye calibration to solve instead for the relationship $Y$ between the calibration object and the end effector, as follows:

$$MY = M'YB \tag{2.9}$$

The advantage of their formulation is that the extrinsic and intrinsic parameters of the camera do not need to be made explicit as it is in classical formulations. This is because $M$ and $M'$ are $3 \times 4$ perspective matrices associated with two positions of the camera with respect to the calibration frame. This new formulation allows a wider range of camera–based sensors to be calibrated with respect to the end effector.

Consequently, their work presents a unified optimal solution to the problem as follows. Since their formulation, shown in Equation 2.9, decomposes into

$$\begin{aligned} n_N &= R_Y n_B \\ (N - I)t_Y &= R_Y t_B - t_N \end{aligned} \tag{2.10}$$

which is of the generic form

$$\begin{aligned} v' &= Rv \\ (K - I)t &= Rp - p' \end{aligned} \tag{2.11}$$

In this case, $R$ and $t$ are the parameters to be estimated (*i.e.*, rotation and translation). In order to estimate these parameters, a known calibration object is required to capture at least two camera motions (*i.e.*, three views). In the general case of $n$ motions, the problem can be solved by minimizing the following positive error functions:

$$f_1(R) = \sum_{i=1}^{n} \|v_i' - Rv_i\|^2 \tag{2.12}$$

and

$$f_2(R,t) = \sum_{i=1}^{n} \|Rp_i - (K_i - I)t - p_i'\|^2 \tag{2.13}$$

By representing rotations by unit quaternions $q = [\ r_0 \quad r_x \quad r_y \quad r_z\ ]^T$, minimization can be achieved in one of two ways:

1. Rotation is estimated first by minimizing $f_1$ using a closed-form solution. Then the minimization of $f_2$ becomes a linear least-squared problem.

2. Rotation and translation can be estimated simultaneously by minimizing $f_1 + f_2$. It is a non-linear problem, but it provides a more stable solution.

Since our system employs inertial navigation specifically to track translations between frames, we are only interested in solving for rotation to determine the relationship between two sensors mounted rigidly to the same object. Then to minimize Equation 2.12 Horaud & Dornaika write:

$$\begin{aligned} Rv_i &= q * v_i * \bar{q} \\ \|v_i' - q * v_i * \bar{q}\|^2 &= \|v_i' - q * v_i * \bar{q}\|^2 \|q\|^2 \\ &= \|v_i' * q - q * v_i\|^2 \\ &= (Q(v_i')q - W(v_i)q)^T (Q(v_i')q - W(v_i)q) \\ &= q^T \Lambda_i q \end{aligned}$$

Here, $\Lambda_i$ is a $4 \times 4$ symmetric matrix:

$$\Lambda_i = (Q(v_i') - W(v_i))^T (Q(v_i') - W(v_i))$$

Then the error function becomes:

$$\begin{aligned} f_1(R) &= f_1(q) \\ &= \textstyle\sum_{i=1}^{n} \| v_i' - q * v_i * \bar{q} \|^2 \\ &= \textstyle\sum_{i=1}^{n} q^T \Lambda_i q \\ &= q^T (\textstyle\sum_{i=1}^{n} \Lambda_i) q \\ &= q^T \Lambda_i q \end{aligned} \tag{2.14}$$

$$Q(r) = \begin{pmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & -r_z & -r_y \\ r_y & r_z & r_0 & -r_x \\ r_z & -r_y & -r_x & r_0 \end{pmatrix} \qquad W(r) = \begin{pmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & r_z & -r_y \\ r_y & -r_z & r_0 & r_x \\ r_z & r_y & -r_x & r_0 \end{pmatrix}$$

Since $\Lambda = \sum_{i=1}^{n} \Lambda_i$, and $q$ is a quaternion, one can minimize $f_1$ as follows:

$$\min_q f_1 = \min_q (q^T \Lambda q + \lambda(1 - q^T q))$$

Lastly, by differentiating this error function with respect to $q$, one may easily find the solution in closed form:

$$\Lambda q = \lambda q$$

The unit quaternion that minimizes $f_1$ is the eigenvector of $\Lambda$ associated with its smallest positive eigenvalue. This closed-form solution was introduced by Horn [34] and Faugeras & Hébert [23]for finding the best rotation between two sets of 3D features.

The hand-eye calibration problem is summarized in Figure 2.1(a) where the desired relationship is represented by $X$ for the classical formulation [88], and $Y$ for the alternative formulation [33]. Extending this to our system, $A$ is the transformation between the calibration frame and the camera, $B$ is the transformation from the INS frame to the world frame, and $X$ is the transformation from the INS frame to the camera frame, as illustrated in 2.1(b).

More recently, Strobl & Hirzinger [81] did more advanced work that employs both, but distinguishes between the two common solutions of the hand eye calibration problem. They presented a physically based metric on the Euclidean group of rigid-body motions for optimal estimation within these two formulations. Additionally, their algorithm aims to reduce the error that would accumulate between sensor positions in older approaches. Their novel metric makes it possible to select the appropriate formulation in relation to the actual system characteristics.

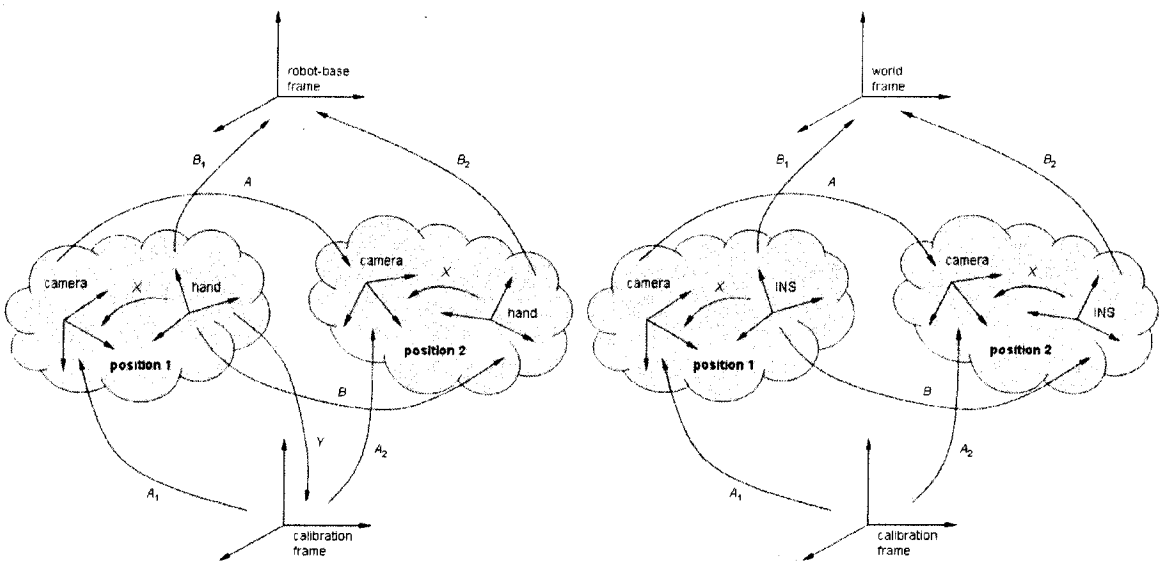



Figure 2.1: General overview of (a) the calibration problem showing two different positions of the hand–eye device, and (b) the transformations required to execute a data registration procedure using our system. Comparing them, it is clear that the calibration method to solve for $X$ proposed by Horaud & Dornaika applies just as well in alternative applications as long as the relationships $A$ and $B$ are known. Figure adapted from [33].

Along with an experimental error model, the metric is used to perform optimal estimation in the form of the maximum likelihood (ML) method where translational and rotational errors are weighted with the particular precision characteristics of the manipulator. The maximum likelihood method selects the formulation for which the probability of the observed data is highest (*i.e.*, for which its incompatibility with the model is minimized). Furthermore, the weights are adapted automatically by the algorithm. While this work does show improved performance compared to existing methods and could potentially apply to our system, it is much more computationally expensive because it must first determine for which formulation to solve, in addition to the solution itself.

## 2.3 Multi-View Data Registration

No matter what hardware and techniques are used in a 3D scanning system, there remains a need for additional alignment between scenes because of error attributed to noise,

inaccuracies in the stereo correspondence or low resolution measurements. Besl & McKay [8] and Chen & Medioni [16], developed the Iterative Closest Point (ICP) algorithm, which has become one of the most prolific techniques for multi-view data registration in the field. Given an initial estimate of their relative transformation, ICP can align multiple sets of 3D geometry once they have been decomposed into sets of points. The algorithm works by finding corresponding points in each set, computing a transform between these points, updating each set with the transform, and iterating until the point sets are sufficiently aligned. The general ICP algorithm has the following steps:

1. Associate corresponding points by the nearest neighbour criterion.
2. Estimating the transformation parameters using a mean square cost function.
3. Transforming the points using the parameters estimated in the previous step.
4. Iterating until the mean square error falls below a specified criterion, or the iteration limit is reached.

The basic idea behind the algorithm is that a closest point determination reduces the distance for each point individually, and consequently, the average distance between points is reduced because the average of a set of smaller positive numbers is smaller. While the ICP algorithm always converges monotonically to a local minimum with respect to the mean-square distance measure, it can be troublesome when there are outliers involved, and sometimes does not converge to the global solution. Fortunately, its popularity has spawned many variations of this algorithm as faster and more accurate techniques are developed to replace each function.

Consequently, Rusinkiewicz & Levoy [66] reviewed many ICP variants and recorded the methods used for each step in the process. Several variants extended the original algorithm from Besl & McKay and Chen & Medioni to include additional steps for point weighting and rejection. The extended ICP algorithm has the following taxonomy:

1. Selection of some set of points in one or both meshes.
2. Matching these points to samples in the other mesh.
3. Weighting the corresponding pairs appropriately.
4. Rejecting certain pairs based on some criterion.
5. Assigning an error measure based on the point pairs.

6. Minimizing the error metric.

The strategies for the selection of point pairs include: always using all of the available points, uniformly subsampling the points, random sampling the points at every iteration, or choosing those points with high intensity colour/intensity gradients. Rusinkiewicz & Levoy also proposed a strategy of their own such that points are selected in such a way that the distribution of normals among them is as broad as possible. They observed that for some scenes small features of the model are vital to determining the correct alignment, but they do not get equal representation in strategies like random, or uniform sampling. Normal-space sampling can be achieved by bucketing points according to the position of their normal in angular space, then sampling as uniformly as possible across the buckets.

There are also several methods to choose from for finding correspondence between point sets. The most common choice is locating the closest point in the other mesh, which can be accelerated using a K-D tree and/or closest point caching. Alternatively, "normal shooting" is the process of finding the intersection of a ray starting from a source point, in the direction of the its normal, with a reference surface. "Reverse calibration" projects the source point onto the destination mesh, from the point of view of the destination meshs range camera. The source point can also be projected onto the destination mesh before performing a search in the destination range image using any of the previous methods. The "closest compatible point" or "compatibility of normals" option ensures that the normals of corresponding points are within some angular threshold of one another. Compatibility metrics based on colour have also been explored

Assigning of weights to each corresponding point pair computed in earlier steps can be done in one several ways as well. The default method is using constant weight for each pair, or doing nothing. Alternatively, lower weights can be assigned to points that are further apart. Weighting can also be done based on the compatibility of the normals of the point pairs, or by colour matching. Lastly, it can be based on the expected effect of scanner noise on a point pair or the uncertainty in the error metric.

Rejecting certain pairs is closely related to assigning weights to corresponding pairs. The purpose of this step is usually to eliminate outliers, which may have a large effect on the error metric. Some common strategies proposed for this step include rejection of points which are more than some distance apart, rejection of the worst $n\%$ of pairs based on a similar metric, or rejection of pairs whose point-to-point distance is greater than some multiple of the standard deviation of distances. Another option is to reject pairs that are not consistent with neighbouring pairs, assuming surfaces move rigidly

which classifies two correspondences $(p_1, q_1)$ and $(p_2, q_2)$ as inconsistent if $|Dist(p_1, p_2) - Dist(q_1, q_2)|$ is greater than some threshold. Lastly, rejection of pairs containing points on mesh boundaries is especially useful for avoiding erroneous pairings.

There are also a variety of error metrics available to use with ICP. Firstly, the sum of squared distances between corresponding points is often used because there exist closed form solutions to this type of metric. Solution methods based on singular value decomposition, quaternions, orthonormal matrices, and dual quaternions have also been proposed. It is also possible to use any of the previously mentioned metrics while taking into account both the distance between points and the difference in colours. A "point-to-plane" technique determines the sum of squared distances from each source point to the plane containing the destination point and oriented perpendicular to the destination normal. However, no closed–form solutions are available in this case so the least squares equations must be solved using a generic non-linear method or by linearizing the problem (assuming incremental rotations are small).

There are several ways to formulate the minimization of an error metric. The first option is to generate a set of corresponding points using the current transformation at each iteration, and finding a new transformation that minimizes the error metric. One alternative to that is to perform the iterative minimization starting with several perturbations in the initial conditions, then selecting the best result. This motivation behind this technique is the fact that ICP can stop prematurely when the error metric converges to a local minimum, which may be a false match. Another technique involves performing the iterative minimization using randomly selected subsets of points, then picking the best result using a robust least median of squares metric. Lastly, there is the option of performing a stochastic search for the best transform using simulated annealing.

The information summarized in the review of Rusinkiewicz & Levoy allows us to select an ICP variant that is tailor made for our system. Since our implementation is restricted to the use of point clouds, it limits our options to point and normal-based methods because mesh surfaces are not available. It should also include a strong rejection method to eliminate the outliers that are expected to occur when using a stereo camera. Examples of specific methods and algorithms for each stage are available in [66] and related references.

## 2.4 3D Scanning Systems

Recent advances in hardware and algorithms for combining range images have helped to popularize and expand the field of 3D scanning. One of the most notable systems in recent years is the digital Michaelangelo project by Levoy et al. [44]. Their system uses a combination of custom built laser triangulation rangefinders, laser time-of-flight rangefinders, digital still cameras, support gantries and a suite of software for acquiring, aligning, and merging scanned data. Overall, the equipment weighed a total of 4 tons.

A typical scanning procedure begins with an operator manually teaching the scan head a sequence of planned motions, and setting the limits of the volume to be scanned prior to running an automated script that reproduces the same motions. Alignment of the scans requires a coarse manual estimate done by an operator, followed by a modified iterative closest point algorithm [8, 17] to refine the local alignment, and concludes with a global alignment using a global relaxation algorithm [62]. Lastly, surface reconstruction is done using a voxel-based method [20].

The main contributions of the Michaelangelo project were to firstly push the boundaries of state-of-the-art scanning, and to develop new methods to capture and manage extremely large models – including a novel compressed file format to store range maps. The most obvious disadvantage of such an elaborate system is that it is difficult and expensive to transport. Furthermore, due to its size, assembly and operation under field conditions (*i.e.*, non-laboratory) has been described as "highly prohibitive, and a potential liability" [44]. It is also clear that many operations in the system are not automated, and require manual intervention from an operator.

Van Den Hengel et al. [89] developed an interactive method called VideoTrace to generate 3D surface models of objects from video. This system works by the principle of structure from motion. It is a computer vision technique that builds from video a sparse set of 3D scene points and the camera parameters that describe the relationship between the camera and the scene. In contrast to the previous implementation, VideoTrace represents a lower cost hand held scanning system, but it requires significant user input, and it cannot capture the geometry of finer surface details.

The procedure typically begins by representing each image in terms of "superpixels" [63] instead of raw pixels in order to accelerate any image operations (*i.e.*, tracing, edge detection, clustering). A user then manually traces out the boundaries of any surfaces in the image to produce polygonal faces, and multiple frames are used to refine the clustering of superpixels in those faces. It also includes a toolkit for defining 2D and 3D

Non-Uniform Rational B-Splines (NURBS) that allow users to trace 2D and 3D curves where the problem of curve estimation is posed as a graph-based optimization. NURBS curves can be restricted to lie in a plane, or used to generate NURBS surfaces.

This system uses a single piece of inexpensive commercial hardware, which qualifies it as an inexpensive mobile 3D scanning platform. It also claims to combine the benefits of automatic and manual modelling systems because users can control which parts of a scene are modeled, and to what detail. What the system lacks primarily is the ability to localize the camera, and hence objects, in the real world.

A mobile scanning platform similar to ours is the instant Scene Modeler (iSM) by Se & Jasiobedzki [73]. It is a 3D imaging system that creates models using only a single hand-held stereo camera; it includes no additional hardware components other than a computer. They use Bumblebee stereo camera from Point Grey Research at 640 × 480 image resolution with a maximum capture rate of 15 frames per second (FPS). Their software will run on any PC equipped with IEEE 1394a support, which is necessary to interface with the stereo camera.

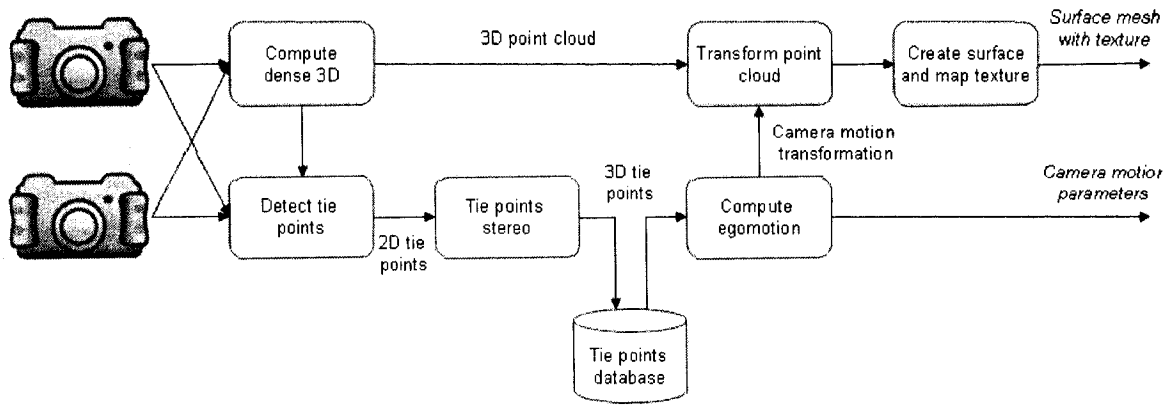



Figure 2.2: Flow chart of the high level architecture of the iSM from Se & Jasiobedzki. Figure adapted from [73].

The high-level architecture (HLA) of their system is outlined in Figure 2.2. Images are captured by the stereo camera and dense stereo disparity is computed for each stereo pair using the PGR Triclops library to obtain 3D data using some known camera calibration. Instead of using an extra sensor to track camera motion like we do, their system employs a high-level set of visual features called Scale Invariant Feature Transform (SIFT) [45] as the anchors for the computation of the camera pose. SIFT features are invariant to image translation, scaling, rotation, and partially invariant to illumination

changes and affine or 3D projection. The recovered camera motion is used to fuse the 3D data before it is converted to surface meshes textured using the colour camera images.

Typically, any given image will have hundreds of SIFT features whose 3D positions $(x, y, z)$ are subsequently computed using the following equations:

$$X = \frac{(u-u_0)I}{d} \quad Y = \frac{(v_0-v)I}{d} \quad Z = \frac{fI}{d} \tag{2.15}$$

Here, $(u, v, d)$ are the SIFT image location and disparity, $(u_0, v_0)$ are the image centre coordinates, $I$ is the baseline distance of the camera and $f$ is the focal length of the camera. Afterwards, a Simultaneous Localization and Mapping (SLAM) approach uses the SIFT features to localize the view and simultaneously build a database map [74] to reduce accumulation error associated with frame to frame matching.

Lastly, a voxel-based method [65] is used to generate triangular meshes, their associated normals, and to fill up holes. Each triangle is textured using colour image information from several views in which the camera's viewing axis is as close to normal to the triangle in question as possible.

Labrie & Hébert [42] develop a system that uses a camera combined with an inertial sensor, as opposed to purely feature based algorithms. Its primary goal is to exploit the orientation obtained from the inertial sensor to accelerate online registration between images. The main contribution of this work is to improve upon a system [53] that registers triplets of consecutive images linked together in a chain. Using such subsets of images allows this implementation to estimate camera positions in the real world by solving systems of linear equations. This implementation is similar to ours in the way they match consecutive images, and their use of an inertial sensor to the orientation of the camera with respect to an intrinsic reference. However, their use of the INS is limited entirely to orientation data, choosing instead to use feature-based methods to calculate the motion of the camera.

The first task in this implementation is the selection of feature points in each image which correspond to physical points visible in multiple frames. These points must have a "signature neighbourhood" that characterizes them uniquely (*e.g.*, edge contour points, or high curvature areas). These features are subsequently localized at subpixel resolution. Typically, the number of feature points is limited to a maximum of 500 per image, and only these sparse feature points are used in the final mesh(as opposed to a dense set of points). Matching points are determined by evaluating the similarity between the image regions around feature points.

The inertial sensor provides the orientation of the camera with respect to the world

wherever and image is captured. They obtain the orientation of each snapshot by compositing the INS data at each camera position with the inverse orientation at the first position in the sequence as follows:

$$R_{1i} = R_i R_1^T \tag{2.16}$$

$R_i$ represents the orientation from the inertial sensor that is associated with image $i$, and $R_{1i}$ is the relative orientation between images $i$ and 1. In order to determine accurate feature based translation vectors within the image triplets, they require feature points that appear in all three images to minimize the projection error of 3D points $X$ triangulated from the first two images. Solving the two linear equations below, relates the image triplets so that they all share the same coordinate system.

$$T_{13} = \alpha X'_x - x X'_z + u_o X'_z \tag{2.17}$$

$$T_{13} = \beta X'_y - y X'_z + v_o X'_z \tag{2.18}$$

In these formulas, $(x, y)$ is a feature point, $X' = R_{13}X$, and $\alpha$, $\beta$ and $(u_0, v_0)$ are the intrinsic scale factors and the principal point, respectively. Multi-view registration is ultimately done using a RANSAC type algorithm [25] because of the relatively small number of features that are retained from each frame.

While their work only makes use of the inertial sensor to track orientation, it does indicate that as the hardware improves, these types of sensors will become increasingly useful in such applications. It also implies that, for the time being, this type of inertial tracking still benefits more from vision based methods to produce good results over longer sequences.

Pollefeys et al. [60] worked on a more ambitious system that tries to incorporate data from a Global Positioning System (GPS) as well as an inertial sensor to reconstruct models of entire cities from video. While the GPS and INS data is used for camera pose estimation with a Kalman filter, if it ever fails or is unavailable for any reason, the system falls back on structure from motion algorithms. Specifically, the system has a module to track up to 1000 salient image features, and they propose an novel extension of the KLT tracker that estimates the gain of the camera [40].

Their system operates at 30 frames per second and a resolution of $512 \times 384$ pixels. In order to process the amounts of data produced, their system operated on a computer that featured a dual core AMD Opteron processor at 2.4 Ghz and an nVidia GeForce 8800 series GPU, which amounts to a standard PC today. The high level architecture

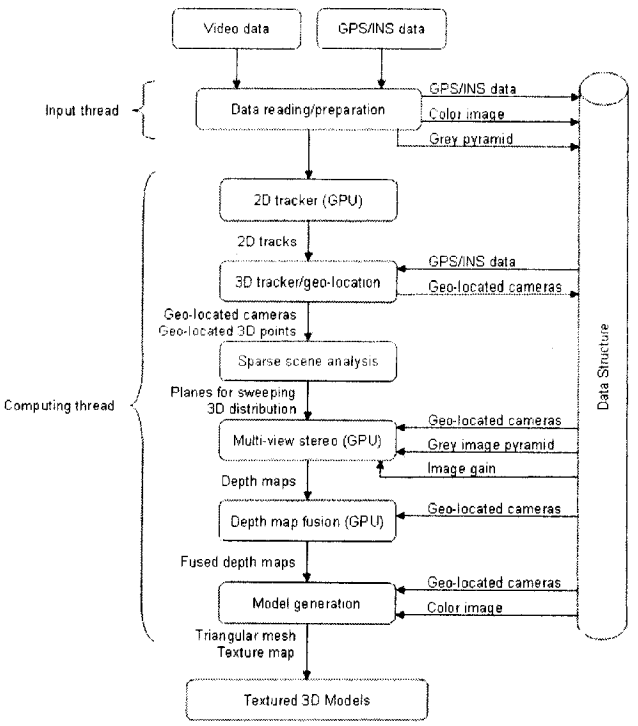



Figure 2.3: Flow-chart of the high level architecture of an urban modeling system from Pollefeys et al. Figure adapted from [60].

of their system is illustrated in Figure 2.3 indicating the processing pipeline and data structure utilized by their system.

The depth maps employed in their procedure are generated using an extended plane sweeping stereo technique on video from a single camera and depth fusion is performed at half-resolution at 4.88 ms per frame. Lastly, a mesh is generated for each fused depth map, and then textured using JPEG image information – an easy to use technique.

An important contribution of their work is the development of visual techniques that could deal with the large brightness variation of outdoor scenes (*i.e.*, tracking and compensating for gain changes). Additionally, their technique tries to leverage the processing power of modern graphics cards which are becoming increasingly powerful. They also tend to focus their effort on exploiting redundant data (each element is typically seen in tens of views) rather than trying to determine and use the minimum amount of data needed. It is a uniquely geo-localized method for use in urban areas, while others like ours operate in a much smaller environment.

## 2.5 Summary

In this chapter, we discussed some of the more pertinent techniques and algorithms that are used in the field of 3D scanning. The stereo correspondence algorithm in particular plays an important role in the final quality of the model because it determines the accuracy of the distance measurement from the camera to a point on the target. Furthermore, differences in the implementation of such an algorithm can vary the results, and a review of existing methods compares our implementation to other methods.

Likewise, the point matching algorithm plays an important role in the registration of multiple views into proper alignment. Section 2.3 discusses many variants of the iterative closest point algorithm which can be applied with varying degrees of success to any given data set.

Some fundamentals of stereo vision were discussed in Section 2.1, along with a taxonomy that may be used to characterize stereo correspondence algorithms. Many approaches have opted for the fast, reliable sum of absolute intensity differences matching cost measure, with a window–based aggregator. However, there was no overwhelming consensus about which disparity computation is the most robust. Section 2.2 presented common methods used for calibrating a device mounted to an end effector. It is a necessary step that determines the relationship between a sensor and the device to which it is mounted, and enables one to track the sensor's orientation in the world. Furthermore, Section 2.4 gives an overview of other scanning systems currently available. They range from those that rely heavily on image feature based approaches to determine camera motion, to those that use external sensors to localize the camera in the world, as well as some that combine both techniques. We will show in this thesis that the system we have assembled to digitize 3D subjects is as capable as the larger, more resolute laser scanners available, and does so in a mobile configuration.

# Chapter 3

# System Calibration

In this chapter we present the hand-eye calibration component of our 3D scanning system which conforms with the 3D model acquisition pipeline described by Bernardini & Rushmeier [6]. In Section 3.1 we briefly discuss the hardware components and configurations employed by our system, mainly consisting of a commercial stereo camera and inertial navigation system. A standard PC or laptop is employed as a host and is used to acquire sensor data. Section 3.2 outlines the specifications and motivation behind the calibration object used in our system. A calibration object provides us with a subject with known dimensions in the scene being viewed by the camera. Section 3.3 details the solution for the hand-eye calibration problem, and a normal least–squares (LSQ) approximation of the relationship between the camera and the scene. Our results and a summary of the topics of this chapter are discussed in Section 3.4 and Section 3.5, respectively.

## 3.1 Hardware

The system consists primarily of two devices mounted to a rigid platform such as a mobile robot, a helmet, or a frame. The first device, shown in Figure 3.1, is a Bumblebee stereo camera from Point Grey Research, which utilizes two individual progressive scan colour CCD cameras. Each camera includes pre–calibration for lens distortions and camera misalignments by factory standard, including automatic gain and shutter control with an adjustable frame rate and independent control of CCD gains for image intensity balancing. The left and right images are aligned within 0.05 pixel RMS error [59], and the calibration information is pre–loaded onto the camera allowing the software to retrieve image correction information at any time. It is designed for applications such as people

tracking, gesture recognition, mobile robotics and other computer vision applications. Detailed specifications for this device are enumerated in Appendix A.

In the proposed system, the captured data includes rectified colour video with a corresponding disparity range map for each frame captured at a resolution of $1024 \times 768$ pixels at a rate of up to 15 Hz. Power to the camera is supplied either by a PC host with an appropriate 5–pin IEEE 1394a connector, or a separate battery pack equipped with a voltage regulator when only a 4–pin laptop connector is available in the field. Specifics regarding operating voltage and power consumption for this device is available in Appendix A. A few sample frames of the hand-eye calibration data sequence are shown in Figure 3.2.

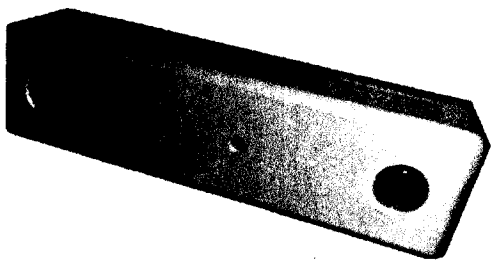
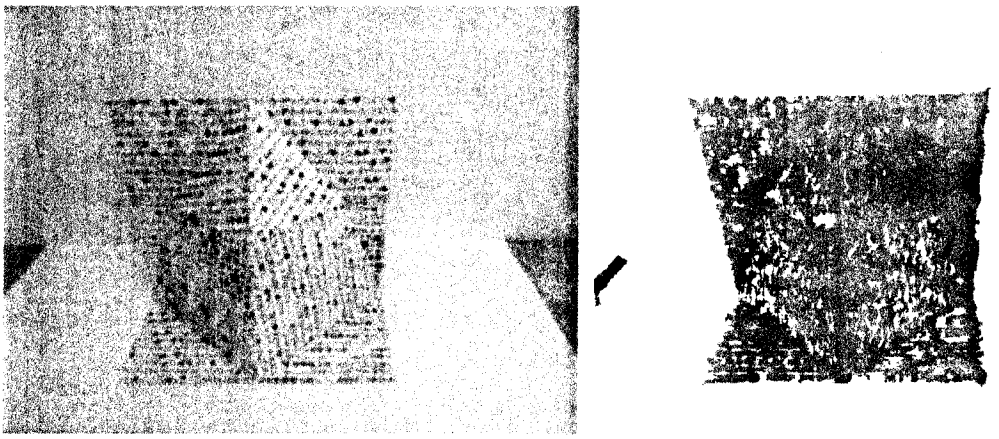


Figure 3.1: Point Grey Bumblebee.

The product package includes the Digiclops standard development kit (SDK) which controls stereo vision camera settings and image acquisition, including device drivers, and a full software library and API for the C/C++ programming environment. Also included is the Triclops SDK which provides real-time disparity imaging using stereo vision technology allowing accurate measurement of the distance to every valid pixel in an image. We selected a commercial solution for data acquisition because the focus of our work is in the data registration process, not in issues relating to stereo vision specifically.

A disparity range map is generated using a stereo correspondence algorithm by exploiting the distance between slightly offset cameras recording the same scene. In Chapter 2 we discussed the fundamentals of stereo vision, and briefly outlined the correspondence algorithm employed by our system. In Summary, the Triclops API utilizes an algorithm that employs a sum of absolute intensity differences matching cost measure, a square window aggregation, and a sub-pixel refinement disparity computation/optimization. There are, however, other post–processing options available in the Triclops API that serve as disparity refinement after correspondence. All available settings, and those that we used for all of the data acquisition and stereo correspondence in our system are summarized in



(a) (b)

Figure 3.2: Sample of (a) a rectified colour image from the reference camera, and (b) its corresponding disparity/range map.

Table 3.1. These settings were selected because they produce low–noise dense disparity maps with few surface holes or outliers.

The second device shown in Figure 3.3, called an MTi, is a "solid–state miniature MEMS gyro-enhanced attitude and heading reference system (AHRS)" available commercially from Xsens Motion Technologies [96]. It has an internal low–power signal processor that computes drift–free orientation, acceleration, rate of turn and earth–magnetic field data. It is designed for applications such as stabilization and control of cameras, robots, vehicles and other equipment. The sensor uses a USB digital interface and so does not require any external power supply when used with a laptop in the field. Detailed specifications for this device are enumerated in Appendix A.

The product package also includes an MT Communication C++ Class that provides a low–level communication protocol, data retrieval and configuration for the tracker as well as a dynamic link library (DLL) with an API for integration with third party development software including Matlab, LabVIEW, VB/VBA, and C/C++. This facilitates the synchronization of data from the INS with the frames of video from the stereo camera. Consequently, the tracker records orientation and acceleration simultaneously at a rate of 120 Hz. All available settings, and those that we used for all of the data acquisition in our system are described in Table 3.2. Here, we use the default recommended settings in most cases because they are the manufacturer recommended settings for normal operation.

| Setting | Enabled | Value | Description |
|---|---|---|---|
| low-pass filter | ✓ | n/a | applies a low-pass filter to each image to reduce noise (*i.e.*, graininess) |
| rectification | ✓ | n/a | correct images for lens distortion and camera misalignment to fit a pinhole camera model |
| edge enhancement | ✓ | n/a | apply a band-pass filter to each image to enhance the intensity of edges |
| edge mask size | ✓ | 9 pixels | set the edge detection window mask for edge enhancement |
| stereo mask size | ✓ | 13 pixels | set the size of the stereo aggregation window |
| disparity range | ✓ | 100–330 pixels | this option sets the maximum valid disparity range of corresponding pixels |
| disparity mapping | n/a | 0–230 (default) | map a disparity range that is smaller than 256 depths during acquisition into a different range (256 depth values maximum) |
| texture validation | – | – | sets a threshold that allows one to tune the texture-based rejection of pixels |
| unique validation | – | – | verifies that the match of a given pixel to its corresponding pixel in the other image is unique enough to be considered the definite correct match based on some threshold |
| back-forth validation | ✓ | n/a | ensure that the left to right and right to left disparity maps agree |
| sub-pixel validation | ✓ | n/a | enables the use of sub pixel resolutions when computing disparity information |
| surface validation | ✓ | 1 pixel | use disparity difference between two pixels to determine whether they can be considered part of the same surface based on some threshold |
| surface size | ✓ | 200 pixels | sets the minimum number of pixels a surface can cover and still be considered valid |
| stereo quality | n/a | standard | selects between a standard and an enhanced stereo qualification |
| rectification quality | n/a | standard | selects between a standard and an enhanced rectification qualification |

Table 3.1: Additional configuration settings for our stereo correspondence application.

Both sensors are mounted to the same rigid frame (*e.g.*, robot, helmet or car) with arbitrary orientations, which results in the hand–eye calibration problem shown in Figure 2.1(b). For our experiments, we used a high density rigid foam block to mount the devices, as in Figure 3.4, so that it may be used as an ordinary hand–held camera. In Summary, it is clear that our setup parallels the classical hand-eye calibration problem encountered when a sensor is mounted to a robot arm as shown in figure Figure 2.1(a).

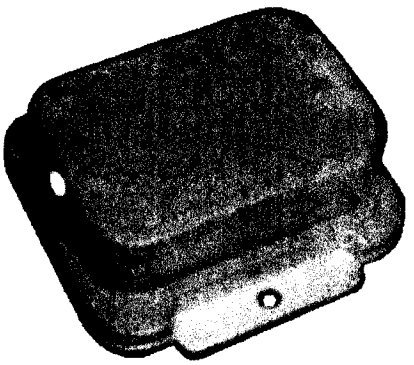


Figure 3.3: Xsens MTi AHRS.

| Setting | Enabled | Value | Description |
|---|---|---|---|
| orientation format | ✓ | rotation matrix | select orientation output representation: *normalized quaternions*, *euler angles* or *rotation matrix* |
| calibrated data | ✓ | n/a | output calibrated 3D acceleration, rate of turn, and magnetic field data in standard units of measurement |
| raw data | – | n/a | output raw binary sensor data |
| coordinate system | n/a | heading | choose a coordinate system of the sensor: *heading* (default), *global*, *object*, or *align* (see Appendix A for further details) |
| filter gain | n/a | 1 Hz (default) | set the "cross over" frequency of the sensor fusion algorithm in Hz (a value of 1 means that all frequency components of the orientation vector >1 Hz will be determined by the rate of turn sensors, and all components <1 Hz will be determined by the accelerometers and magnetometers) |
| filter weighting | n/a | 1 (default) | set how much the sensor data from the magnetometer should be weighted relative to the accelerometer data (a value of 1 indicates that magnetometer data is considered equal to the accelerometer data) |

Table 3.2: Configuration settings for inertial navigation system.

## 3.2 Calibration Target

Before we continue, we would like to introduce a notation for rotation (and homogeneous transformations in general) that was popularized by Craig [19] for the field of robotics. A matrix ${}^{q}_{z}R$ represents the rotation that transforms an object represented in coordinate frame $z$ into coordinate frame $q$. The matrix ${}^{z}_{q}R$ is the inverse of the matrix ${}^{q}_{z}R$, and

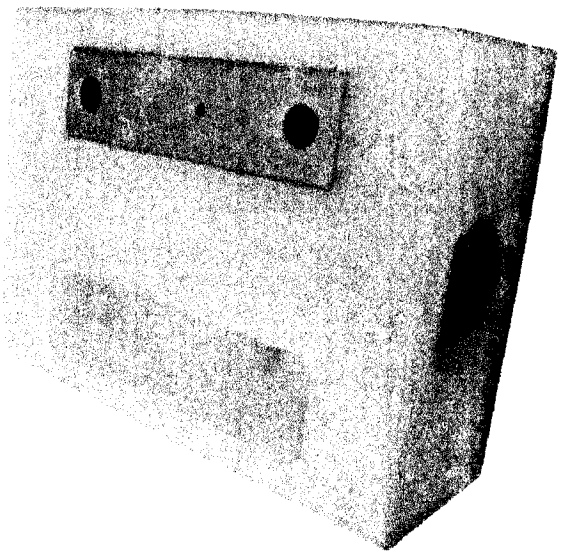


Figure 3.4: The rigid foam block used to mount together our camera and inertial navigation sensor.

if it is a rotation matrix it is also its transpose. We use this notation throughout the remainder of this report with $s$ referring to the INS frame, $w$ to the world, $t$ to the calibration target, and $c$ to the camera.

In order to calibrate our system using a method like that of Horaud & Dornaika [33], it is necessary that we have advance knowledge of the transformation ${}^{c}_{t}R$ between the camera coordinate frame and the coordinate frame of the scene viewed by the camera. We could assign an arbitrary coordinate frame to a random scene, but it would be very difficult to localize any particular point from the scene in that coordinate frame accurately. This problem is avoided by using a known object as a calibration target in your scene during the calibration procedure.

The design of our calibration target was motivated primarily by the least-squares method we use to solve for the transformation ${}^{c}_{t}R$, as well as some limitations in the accuracy of the stereo correspondence along edges. Furthermore, at least three well known features (*e.g.*, vertices or planes) need to be visible from at least three camera positions, assuming six degrees of freedom in the movement of the camera. The dimensions of the calibration target are such that it fits into a cube with a volume of $15 \times 15 \times 15$ cm, and its geometry is chosen such that a minimum of three planes remain visible from any camera position. The redundancy of each additional feature beyond the first three helps to improve the accuracy of the final result. Consequently, our resultant calibration target

it shown in Figure 3.5.

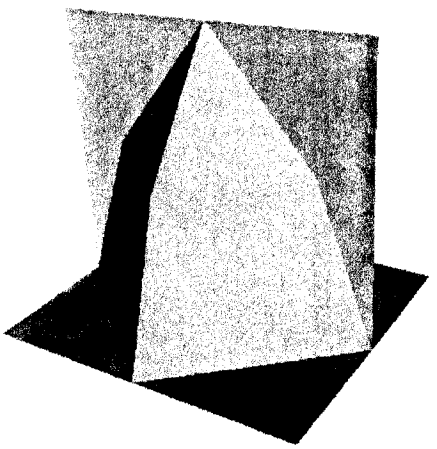


Figure 3.5: Untextured OpenGL model of our stereo calibration target.

However, in order for the stereo correspondence algorithm to function, the surface of the object must be textured. If a surface has a solid colour, too many neighbouring pixels would appear to be the same when attempting to find correspondences between the two images. We chose to simulate a coarse texture by using "Wingdings" (a large font composed of random shapes and symbols available with most word processing software) in order to help the stereo algorithm produce dense disparity maps. Since the correspondence algorithm employed by the Triclops API does not use colour information, the texture was left black and white in order to provide good contrast for data acquisition. It is a good texture for stereo matching because there are sufficient unique symbols that no individual symbol is likely to repeat within the disparity range used for stereo correspondence. The texture was sized so that the symbols are distinguishable in an image when the camera is held approximately 0.5 m away from the target.

This texture would be unsuitable for data acquisition from larger distances because the symbols in the image would become too small as the camera moves away from the target. Multi–resolution patterns that facilitate correspondence from various distances are available (*e.g.*, random shapes of different sizes, or multi–resolution blobs), but unnecessary for our implementation because of the maximum range restriction of our method. A sample of the texture we used is shown in Figure 3.6.

The coordinate frame illustrated in Figure 3.7 was assigned to the calibration target with the origin at the center of the base of the object. Each of the vertices that define the calibration target are localized with respect to this coordinate frame. Similarly, the

Figure 3.6: Sample swatch of the texture applied to the calibration target.

INS and the camera each have their own unique coordinate frames, as shown in Figure 3.8. Any range information produced using the stereo correspondences between images captured by the stereo camera are represented in the coordinate frame of the camera. The coordinate frame of the INS is used primarily to orient the entire device with respect to the earth.

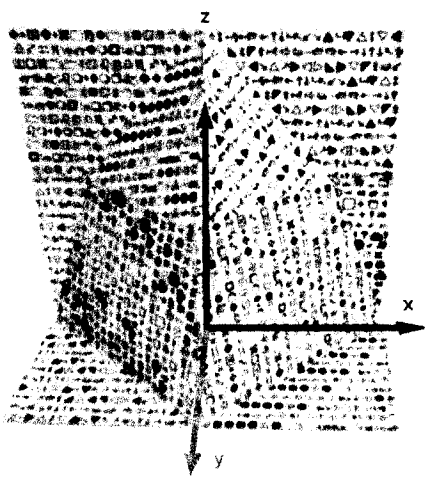



Figure 3.7: Coordinate frame assigned to the calibration target.

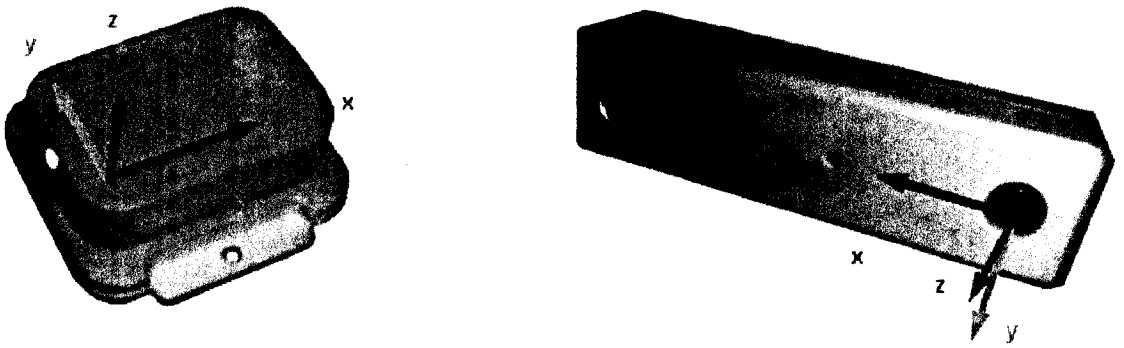



Figure 3.8: Coordinate frames of the stereo camera and the inertial tracker as defined by the manufacturers.

## 3.3 Software and Procedure

With the stereo camera and the INS firmly mounted to a rigid frame, the calibration procedure begins by taking "snapshots" of the calibration target from at least three different positions. This produces range information from the reference camera (*i.e.*, left camera) to the calibration target that can be used to solve for ${}^{t}_{c}R$. It also records the orientation reported by the INS wherever each image is captured, which tells us the relationship ${}^{s}_{w}R$ between the sensor and the world. The world frame in our system refers to the $z$ coordinate along gravity, and the $x$ and $y$ coordinates aligned with the earth's magnetic field. We use six camera positions for the calibration to ensure a more accurate result, and to avoid singularities while solving for the transformations ${}^{t}_{c}R_{i=[1...k]}$, for the general case of $k$ positions. Figure 3.9 illustrates both of these concepts.

At this point some user intervention is required in order to parameterize the intermediate function that determines ${}^{t}_{c}R_{i}$ for each view. This is done through an OpenGL graphic user interface (GUI) created specifically for this task in C++, and is shown in Figure 3.10. The GUI is used to compare each rectified color image captured by the stereo camera against a full-sized metric 3D model of the calibration target stored as a Wavefront OBJ file. The only caveat with the OBJ file is that each vertex must be listed as a member of every face it touches even if its inclusion in a face has no effect on its geometry. An example of this rule is included in Appendix B. Consequently, this allows the calibration target to be changed simply by loading a different OBJ file.

The task of the user in this program is to identify a plane on the 3D model in one part of the GUI, and match it to its image captured by the camera. Using the calibration

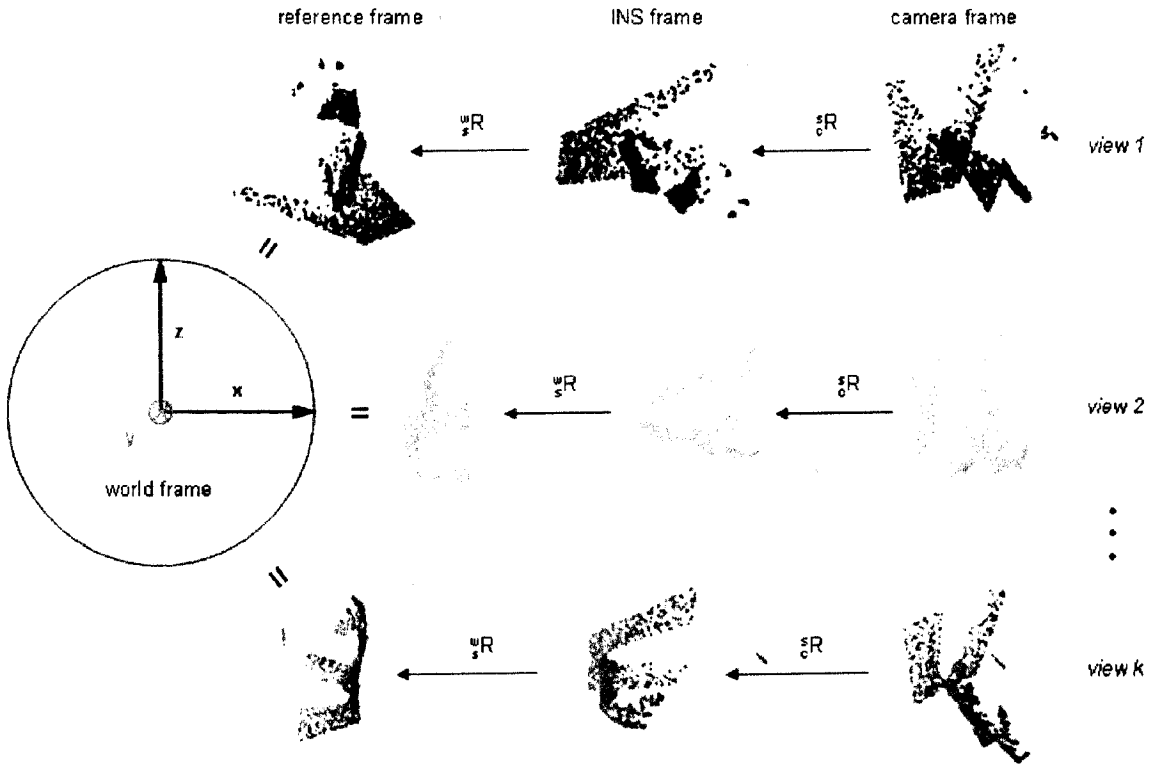



Figure 3.9: Coordinate frame of the world is aligned with gravity and the earth's magnetic field (*i.e.*, arbitrary), and we equate the reference frame with the world frame to be used in all future registration. Transformations from the camera to the inertial navigation sensor, and from the inertial sensor to the world are illustrated here for the general case of $k$ views.

target shown in Figure 3.5, this results in 4–6 planes per frame. The user selects three points in the image that appear on the plane that are used to generate a normal for that plane in the camera coordinate frame. An implicit plane equation is generated using the unit normal and one of the anchor points identified by the user for each plane. The following inequality is then used to determine whether another point in the range map could reasonably lie on that plane by determining its distance from it:

$$\boldsymbol{n} \cdot (\boldsymbol{x}_a - \boldsymbol{x}_t) \leq d \tag{3.1}$$

In this case, $\boldsymbol{n}$ is the unit normal of the plane, $\boldsymbol{x}_a$ is a known anchor point, $\boldsymbol{x}_t$ is the point to test, and $d$ is the threshold distance criterion. Consequently, we consider any point within 3 mm of the plane to be a member. However, if the number of points

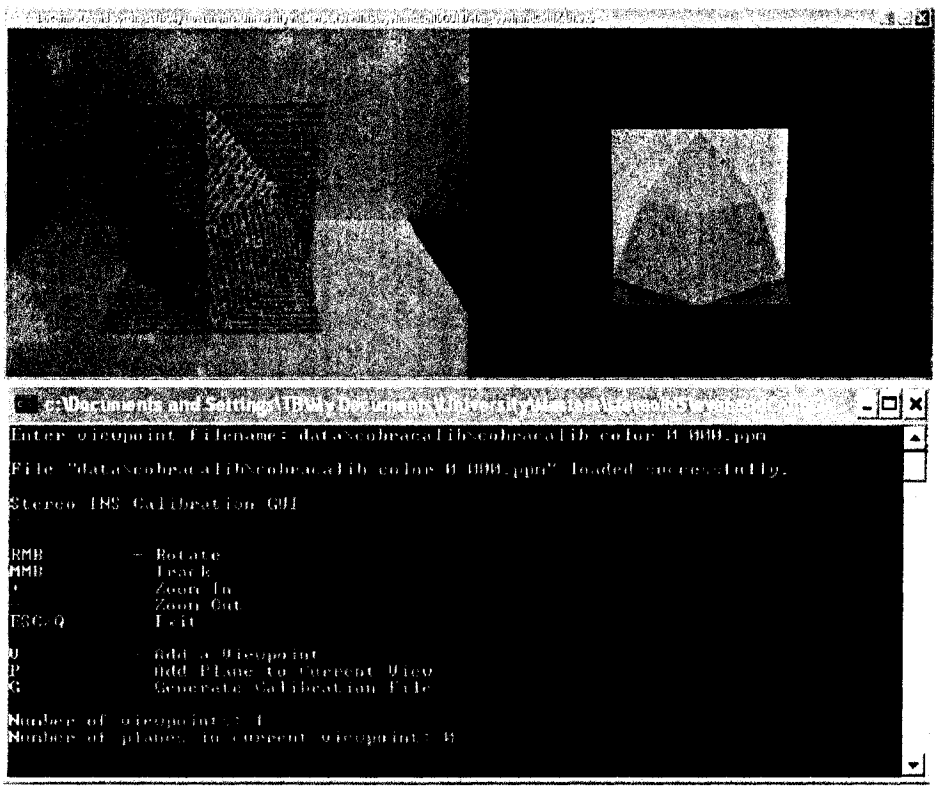



Figure 3.10: The hand-eye calibration graphic user interface. The left frame contains a rectified colour image from the reference camera (*i.e.*, left camera), and the right frame contains an OpenGL model of the calibration object.

that meet this criterion is insufficient (*e.g.*, % of total points), the plane equation is discarded and a new set of inputs is required from the user. Once all memberships have been assigned, a generic plane fit using linear least squares is employed to improve the correctness of our result by determining the best fit for the points that meet the criterion in Equation 3.1.

If $z = Ax + By + C$ is the best fit plane and $\boldsymbol{n} = [\ A \quad B \quad C\ ]$ is its normal, then the problem can be solved using the method of normal equations for $m$ points shown in Equation 3.2. We chose this technique for this task because it is fast and effective for "easy" least squares problems like this one. Its speed comes partly from the fact that it has no confidence mechanism, but mostly because it can be executed in a single pass. The disadvantage of this technique is that it can often fail to converge for more difficult or underdetermined problems (*i.e.*, too few data points), and it is rather susceptible to

roundoff error [61]. In our case, the purpose of the procedure is merely to improve upon a good estimation using a very large set of data points.

$$\begin{bmatrix} \sum_{i=1}^{m} x_i^2 & \sum_{i=1}^{m} x_i y_i & \sum_{i=1}^{m} x_i \\ \sum_{i=1}^{m} x_i y_i & \sum_{i=1}^{m} y_i^2 & \sum_{i=1}^{m} y_i \\ \sum_{i=1}^{m} x_i & \sum_{i=1}^{m} y_i & \sum_{i=1}^{m} 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{m} x_i z_i \\ \sum_{i=1}^{m} y_i z_i \\ \sum_{i=1}^{m} z_i \end{bmatrix} \tag{3.2}$$

We mentioned previously that our procedure requires some set of features to use as reference markers in order to determine this solution. These features are commonly predefined points on the calibration target. However, our primary features are actually the unit normals of the planar surfaces in the scene. The reasoning behind this choice is that the range values along edges are not as accurate as the surfaces themselves due to error in the stereo correspondence algorithm. As a result, the normal of each plane in both camera and calibration target coordinate frames is saved, and exported to file when a sufficient number of views have been analyzed.

Additionally, the system determines all of the unique vertices where three planes intersect. Equation 3.3 is a test to determine whether the intersection of three arbitrary planes is actually a unique point before proceeding.

$$\boldsymbol{n}_1 \cdot (\boldsymbol{n}_2 \times \boldsymbol{n}_3) \neq 0 \tag{3.3}$$

If so, then since $\boldsymbol{n} = [A\ B\ C]$ we can rewrite an implicit plane equation of the form $Ax + By + Cz + d = 0$ to the form $\boldsymbol{n} \cdot \boldsymbol{x} + d = 0$, and the vertex is simply calculated by using Equation 3.4 and exported to file along with the normal information.

$$\boldsymbol{p}_0 = \frac{-d_1(\boldsymbol{n}_2 \times \boldsymbol{n}_3) - d_2(\boldsymbol{n}_3 \times \boldsymbol{n}_1) - d_3(\boldsymbol{n}_1 \times \boldsymbol{n}_2)}{\boldsymbol{n}_1 \cdot (\boldsymbol{n}_2 \times \boldsymbol{n}_3)} \tag{3.4}$$

Unfortunately, this option is limited by the geometry of the calibration target, because there must be at least five unique planes in the frame in order to generate three vertices per frame. Ultimately, using vertex information determined in this way places extra requirements and restrictions on the design of the calibration target. While our calibration supports the use of both normals and vertices as features, our calibration target does not. Consequently, the vertex (*i.e.*, redundant) information is simply ignored because of a lack of unique three-way intersections due to occlusion.

The final transformations ${}^{t}_{c}R_{i=[1\ldots k]}$ are then calculated using a short procedure involving singular value decomposition (SVD). This technique is significantly more robust than using normal equations because it employs confidence values and converges for both

overdetermined and underdetermined systems [61]. The data set for this task (typically half a dozen features) was potentially too small for the method of normal equations to converge. Consequently, we employed the SVD to solve this problem instead because, theoretically, it cannot ever fail. The only disadvantage of the SVD is that it can be significantly slower than the previous solution, but the increased computational load is offset by the relatively tiny size of the data set.

Each design matrix $D$ in Equation 3.5 is constructed using $m$ normals represented in the coordinate frame of the calibration target, and computed from the same viewpoint. Each matrix $y$ in Equation 3.6 is constructed from the same $m$ normals, only as seen in the camera coordinate frame.

$$D_i = \begin{bmatrix} {}^t n_1 x & 0 & 0 & {}^t n_1 y & 0 & 0 & {}^t n_1 z & 0 & 0 \\ 0 & {}^t n_1 x & 0 & 0 & {}^t n_1 y & 0 & 0 & {}^t n_1 z & 0 \\ 0 & 0 & {}^t n_1 x & 0 & 0 & {}^t n_1 y & 0 & 0 & {}^t n_1 z \\ {}^t n_2 x & 0 & 0 & {}^t n_2 y & 0 & 0 & {}^t n_2 z & 0 & 0 \\ 0 & {}^t n_2 x & 0 & 0 & {}^t n_2 y & 0 & 0 & {}^t n_2 z & 0 \\ 0 & 0 & {}^t n_2 x & 0 & 0 & {}^t n_2 y & 0 & 0 & {}^t n_2 z \\ & & & & \vdots & & & & \\ {}^t n_m x & 0 & 0 & {}^t n_m y & 0 & 0 & {}^t n_m z & 0 & 0 \\ 0 & {}^t n_m x & 0 & 0 & {}^t n_m y & 0 & 0 & {}^t n_m z & 0 \\ 0 & 0 & {}^t n_m x & 0 & 0 & {}^t n_m y & 0 & 0 & {}^t n_m z \end{bmatrix} \tag{3.5}$$

$$y_i = \begin{bmatrix} {}^c n_1 x & {}^c n_1 y & {}^c n_1 z & {}^c n_2 x & {}^c n_2 y & {}^c n_2 z & \dots & {}^c n_m x & {}^c n_m y & {}^c n_m z \end{bmatrix}^T \tag{3.6}$$

Now if $w$ is the number of elements in $U$, then the solution to each of the transformations ${}^t_c R'_{i=[1\dots k]}$ is completed using the pseudoinverse ($\frac{1}{S}$) application of SVD as follows:

$$\begin{bmatrix} U & S & V \end{bmatrix} = \text{SVD}(D_i)$$

$$\beta = \begin{bmatrix} \begin{bmatrix} S_{u,v} = 0 \end{bmatrix}^{-1} & \begin{bmatrix} 0_{9,w-9} \end{bmatrix} \end{bmatrix}, \quad \text{if } u \neq v \quad \forall u, v \in \{1 \dots 9\}$$

$${}^t_c R'_i = V \beta U^T y_i, \quad i \in \{1 \dots k\}$$

The method we just described is equivalent to fitting some rotation $R'$ to the data, but there is no guarantee that the resulting $R'$ will be a valid rotation matrix (*i.e.*, an orthonormal matrix). A pseudoinverse is essentially the best estimate of the inverse of

a matrix that is degenerate in some way and a proper inverse does not exist. Since our system maintains a record of dozens of transformations for any given data set, it is imperative to ensure that each of them is valid, and can actually be executed in the real world.

Using a method proposed by Kanatani [37], we fit $R$ to $R'$ by applying SVD to $R'$ and replacing its singular value matrix $S$ with a $3 \times 3$ identity matrix. This process is demonstrated in Equation 3.7 and ensures that the solution to our calibration problem meets all of the criteria for a rotation matrix and Kanatani showed that this result is the rotation matrix closest to $R'$.

$$\begin{bmatrix} U & S & V \end{bmatrix} = \mathrm{SVD}({}^{t}_{c}R'_i)$$
$$ {}^{t}_{c}R_i = V I_{3,3} U^T, \quad i \in \{1 \ldots k\} \tag{3.7}$$

With the relationships ${}^{t}_{c}R_{i=[1\ldots k]}$ and ${}^{s}_{w}R_{i=[1\ldots k]}$ now known for all $k$ views, the fixed relationship ${}^{c}_{s}R$ between the camera coordinate frame and the INS coordinate frame is determined using Horaud & Dornaika's method as described in Chapter 2. Exploiting the parallels, as illustrated in Figure 2.1, between the calibration problem in robotics and our 3D scanner has allowed us to apply their original algorithm in a different field. As mentioned previously, their technique represents rotation using quaternions, and depends on the convenient property that an eigenvector is equivalent to a quaternion with a zero real part. Their technique also includes a solution for the translational relationship ${}^{t}_{c}T$, but it is not needed in our system because any translations that occur during acquisition are relative to previous camera positions, while orientations are absolute.

Until this point we have used the direct cosine matrix notation to represent rotation in our procedure. While any of the tasks we described thus far can be performed using quaternions, we chose to limit their use to the hand–eye calibration because we do much of our data registration using homogeneous transformations, and quaternions are incapable of representing translation.

## 3.4 Results

Our use of normals as calibration features shows some promising results. Visually, the improvement in alignment between viewpoints is obvious when observed in a common coordinate frame (*i.e.*, the first viewpoint). The orientation of the calibration target before and after hand–eye calibration is illustrated in Figure 3.11 for the six viewpoints

we used in the procedure. This is a rather qualitative measure, however, so we propose an alternative quantitative measure to determine approximately how much error still remains to be corrected in subsequent steps of the scanning pipeline.

The quantitative error measure we use is one that cycles through one set of transformations starting from the world coordinate frame to the calibration coordinate frame, then returns along a parallel set of inverse transformations. This action is summarized in Equation 3.8 for the general case of $k$ viewpoints, and using the first frame as the starting point. We chose the first image to be the reference frame because it is simply convenient to compare all other frames against the first one in the sequence.

Furthermore, the first image is generally also the starting position of the camera, Now, if all of the transformations in the loop are true rotation matrices (*i.e.,* a real square matrix whose transpose is its inverse and whose determinant is 1), and assuming they are all exactly correct, then the result of the transformation loop would be a $3 \times 3$ identity matrix $I_{3,3}$. If this is not the case, then there is some error in the rotation that can be quantified.

$$R_\epsilon = {}^{w}_{s}R_i \, {}^{s}_{c}R_i \, {}^{c}_{t}R_i \, {}^{t}_{c}R_1 \, {}^{c}_{s}R_1 \, {}^{s}_{w}R_1, \qquad i = \{1 \ldots k\} \tag{3.8}$$

Applying this metric to a set of viewpoint combinations, we can represent the error present in each loop by individual Euler angles (*i.e.,* roll, pitch and yaw), or a single arc length across the surface of a sphere. We do this by first converting the rotation matrix $R$ into quaternion notation using Equation 3.9; this vastly simplifies the computation of the absolute error values.

$$q = \begin{bmatrix} r_0 \\ r_x \\ r_y \\ r_z \end{bmatrix} = \tfrac{1}{2}\mathrm{Re}\left\{ \begin{bmatrix} \sqrt{1 + R_{1,1} + R_{2,2} + R_{3,3}} \\ (\sqrt{1 + R_{1,1} - R_{2,2} - R_{3,3}})(\mathrm{sgn}(R_{2,3} - R_{3,2})) \\ (\sqrt{1 - R_{1,1} + R_{2,2} - R_{3,3}})(\mathrm{sgn}(R_{3,1} - R_{1,3})) \\ (\sqrt{1 - R_{1,1} - R_{2,2} + R_{3,3}})(\mathrm{sgn}(R_{1,2} - R_{2,1})) \end{bmatrix} \right\} \tag{3.9}$$

$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases}$$

Then computing arc length and Euler angles is accomplished using Equation 3.10 and Equation 3.11 respectively.

$$\epsilon = 2\arccos(r_0) \tag{3.10}$$

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan(\frac{2(r_0 r_x + r_y r_z)}{1-2(r_x^2+r_y^2)}) \\ \arcsin(2(r_0 r_y - r_z r_x)) \\ \arctan(\frac{2(r_0 r_z + r_x r_y)}{1-2(r_y^2+r_z^2)}) \end{bmatrix} \tag{3.11}$$

| | | Angular Errors | | |
|---|---|---|---|---|
| **Loop** | **Arc ($\epsilon$)** | **Roll ($\phi$)** | **Pitch ($\theta$)** | **Yaw ($\psi$)** |
| $1 \leftarrow 2$ | 2.5995° | 0.5458° | -2.5040° | 0.4235° |
| $1 \leftarrow 3$ | 2.9149° | -1.2117° | 0.0554° | -2.6512° |
| $1 \leftarrow 4$ | 1.4537° | -0.2117° | 0.2073° | 1.4228° |
| $1 \leftarrow 5$ | 4.1333° | -0.6943° | -0.4765° | 4.0496° |
| $1 \leftarrow 6$ | 4.7804° | -1.0641° | -1.3002° | -4.4636° |
| $2 \leftarrow 3$ | 4.3436° | -1.9405° | 2.5027° | -3.0160° |
| $2 \leftarrow 4$ | 2.9867° | -0.7220° | 2.7258° | 0.9677° |
| $2 \leftarrow 5$ | 4.3126° | -1.0904° | 2.0460° | 3.6172° |
| $2 \leftarrow 6$ | 5.3175° | -1.8278° | 1.1316° | -4.8820° |
| $3 \leftarrow 4$ | 4.2060° | 1.0043° | 0.0699° | 4.0844° |
| $3 \leftarrow 5$ | 6.7093° | 0.5680° | -0.6409° | 6.6513° |
| $3 \leftarrow 6$ | 2.2012° | 0.1354° | -1.3145° | -1.7619° |
| $4 \leftarrow 5$ | 2.7589° | -0.4870° | -0.6962° | 2.6278° |
| $4 \leftarrow 6$ | 6.1488° | -0.8374° | -1.4805° | -5.8982° |
| $5 \leftarrow 6$ | 8.5541° | -0.4589° | -0.7464° | -8.5062° |
| median | 4.206° | -0.6943° | -0.4765° | 0.4235° |
| $\mu$ | 4.2280° | -0.5528° | -0.0280° | -0.4890° |
| $\sigma$ | 1.8251° | 0.8229° | 1.4778° | 4.2190° |

Table 3.3: Quantitative measures of hand–eye calibration error across each loop.

Our calibration errors for each of the $\frac{n(n-1)}{2}$ unique loop combinations are summarized in Table 3.3. In general, the errors of the loops that start at the first frame are most pertinent because we selected it as our reference, but we include the remaining loop combinations to ensure the validity of the sensor data. For our purposes, arc length refers

to the magnitude of rotation around the some single axis that will align two coordinate frames, and it is our measure of choice. Comparing the results for two different loops illustrates perfectly why arc length is the superior error measure. In general, you cannot easily determine which loop contains less error just by examining the Euler angles. On the other hand, arc length provides a single ranking that immediately indicates the correctness of the loop. However, the Euler angles may be used in a complementary capacity to identify trends within each individual component of rotation (*e.g.*, $\psi$ tends to fluctuate more).

Of note is that the error may be contributed by ${}^{c}_{s}R$, ${}^{t}_{c}R$ or both because these values are computed during calibration, while ${}^{s}_{w}R$ is a measured value. The amount of error observed can depend on the resolution of the range data, the accuracy of the stereo correspondence, and even the diligence with which the calibration procedure is executed (*e.g.*, number of viewpoints used, number of redundant features identified).

Referring back to Figure 3.11, the first box portrays the reference frame, which remains static during all operations. Each subsequent box illustrates the same comparisons as those in Table 3.3, before and after hand–eye calibration. When reviewing these results, it is obvious that the two point clouds in each box are not in exactly the same position. However, this is not relevant in the context of the hand–eye calibration for our system, because our solution focuses solely on rotation. Each set of range data in the figure is median–centered against the reference so that they are more easily compared visually. The pertinent information is the relative orientation of the two point clouds (*i.e.*, how parallel they appear).

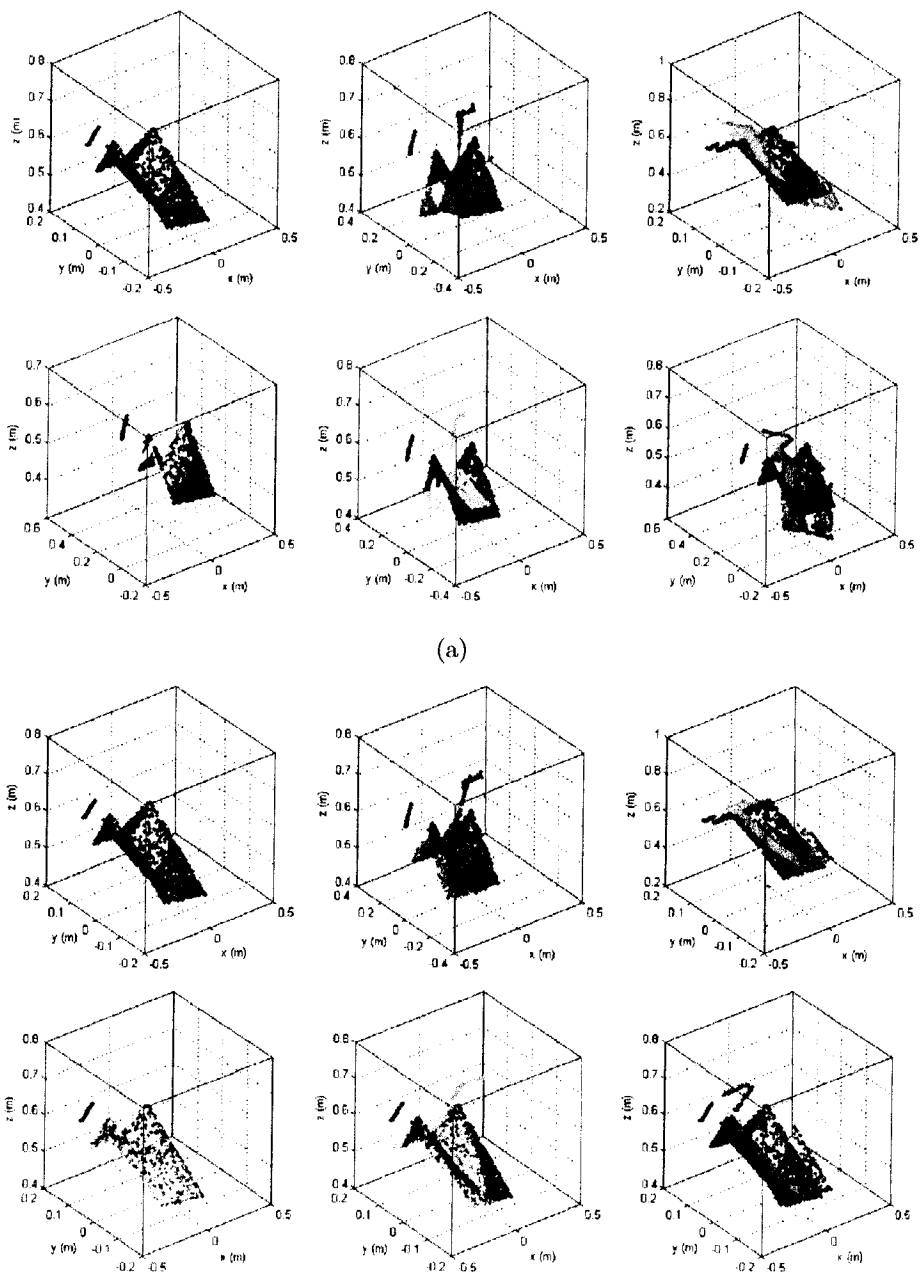
## 3.5 Summary

In this chapter, we discussed the hardware, settings and procedures required to calibrate our 3D scanning system. Section 3.1 discusses the specifications and settings of the commercial hardware we used in our system. Stereo vision is a proven technology that has been around for a long time, but the MEMS technology needed to create compact inertial navigation systems is a much more recent development. As a result, the use of INS devices for 3D scanning applications is still a relatively new field, but it is apparent that our setup parallels the classical hand-eye calibration problem encountered in robotics.

Section 3.2 outlines the need for a calibration target in a camera based scanning system. It also discusses the specifications of our calibration target and the motivation behind its final design. Section 3.3 describes the procedure used to calibrate our system

in depth, our choice of reference features, and introduces the reader to our calibration user interface.

Lastly, Section 3.4 reviewed the results of the calibration, offering both qualitative and quantitative results that confirm its effectiveness. Although some error is present, it is insignificant enough that it should have no effect on subsequent stages of the scanning pipeline.



(a)

(b)

Figure 3.11: Qualitative comparison of orientation between sets of range data (a) before hand–eye calibration, and (b) after hand–eye calibration. The first boxes contain the reference set (shown in blue), and ensuing boxes illustrate the same comparisons as the first five rows of Table 3.3. Calibration does *not* correct the translation error seen here.

# Chapter 4

# Data Registration

In this chapter we present the tools and procedures our 3D scanning system employs for building its 3D models. In Section 4.1 we outline the methods and settings used to initialize both the system and the data for use in subsequent steps, and Section 4.2 outlines the technique to estimate vertex normals.

The quality of the captured sensor data plays a role in the accuracy of the reconstruction of the scenes we scan, and the ease with which our alignment algorithm converges. Camera data is collected at a rate of 1.82 Hz at a resolution of $1024 \times 768$ pixels. The inertial tracker reports orientation and acceleration information at a rate of 102 Hz. The actual process of capturing a data sequence is as easy as using a hand–held video camera, but the user should avoid making sharp motions, or moving too quickly. The reasons for this are to avoid "spiking" accelerometer data and image blur, and to ensure good overlap between consecutive frames.

Nonetheless, we make efforts during our registration to condition the data from both sources with filtering and digital processing to reduce background noise. For example, we aim to locate and remove the ground plane and any outlying points from the stereo data, and to reduce drift and high frequency noise in the acceleration data. These processes are described in detail in Section 4.3 and Section 4.4, respectively.

Section 4.5 describes in detail the implementation of ICP used by our system to align vertex data, and Section 4.6 outlines additional processing that is designed to increase the chance of convergence for data captured using a wide range of motion, including our "similarity grouping" scheme. Section 4.7 details the method we use to generate a surface mesh for the final model, and the way in which it is textured. Lastly, Section 4.8 summarizes the work presented in this chapter.

## 4.1 Initialization

The initialization routine for our registration module first merges the orientation data obtained from the stereo camera and the navigation sensor. Since the INS sensor is rigidly mounted to the same capture platform as the stereo camera (*e.g.*, a wearable system, or mobile robot) it will provide the relationship between the capture platform and the world because the coordinate frame of the INS device is synonymous to the coordinate frame of the hand held apparatus. All vertices and their normals, estimated using the technique in Section 4.2, are transformed from their initial camera coordinate system to the INS coordinate frame, and finally into a world coordinate frame. We apply a compound transformation ${}^{w}_{s}R_{i=[1...k]}{}^{s}_{c}R$ to each of the $k$ sets of vertices in order to represent them and their normals in a common coordinate frame where the $z$ coordinate represents vertical height. These matrices are known: ${}^{s}_{c}R$ is the inverse of our calibration matrix, and ${}^{w}_{s}R_{i=[1...k]}$ are the inverses of the measurements provided by the inertial tracker.

In addition, upon initialization our registration module expects a handful of parameters that are used to register the data, and are summarized in Table 4.1. We refer back to these parameters several times in this chapter.

| Setting | Value | Description |
|---|---|---|
| downsampling factor | 100 | selects the factor $n$ with which to downsample each point cloud by keeping every $n^{th}$ sample, starting with the first |
| ground rejection range | 3.0 cm | sets the width of the rejection range (approximate) to use when removing the ground plane from a point cloud; is equal to twice the resolution of the distribution of $z$ coordinates (see Section 4.3) |
| ICP error threshold | 3.0 mm | sets the minimum point-to-point distance criterion two point clouds must meet in order for their relative alignment to be considered correct (see Section 4.5) |
| group cutoff angle | 20° | sets the rotational threshold that is used to determine whether an image was taken from a similar camera position as another (see Section 4.6) |
| escape condition | 10 | determines the maximum number of iterations our heuristics go through at each stage prior to moving onto the next |

Table 4.1: Typical parameters for our data registration algorithm.

Lastly, colour information for each vertex is extracted from the rectified colour image

captured by the reference camera. Since each vertex corresponds to a certain pixel in the disparity map, the same pixel in the rectified colour image corresponds to the colour of that vertex. Our system produces textures for the reconstructed model by interpolating vertex colour information across each polygon. More vertices result in smaller polygons, which produces a higher resolution texture, and more realistic results.

## 4.2 Normal Estimation

After acquisition, our system estimates a set of vertex normals for the points computed by the stereo correspondence. These normals are required for several tasks including subsampling, grouping and ICP. Since a point cloud is generated from a 2D disparity image, we exploit the fact that any neighbouring pixels in the image will also be neighbours when converted into 3D Cartesian coordinates. Subsequently, we use a window–based method over the image to determine the normal at each pixel, as shown in Figure 4.1.

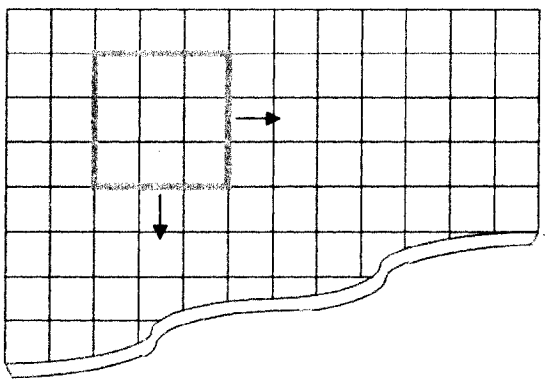
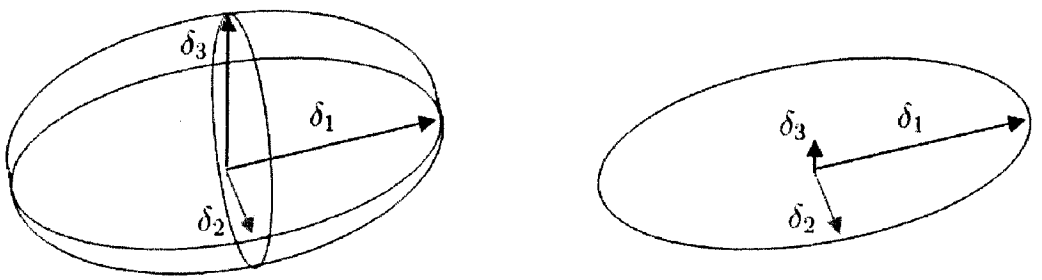


Figure 4.1: Example sliding window used for normal estimation at valid pixels.

Essentially, our initialization routine performs a local plane fitting in the neighbourhood centered around each valid pixel. A neighbourhood is defined by the size of the window; we use a window with a size of $5 \times 5$ pixels. Then using each of the points located at the pixels inside the window, a $3 \times 3$ covariance matrix is constructed according to Equation 4.1 for $n$ valid pixels in each window.

$$\sigma_{xyz} = \begin{bmatrix} \sum x^2 & \sum xy & \sum xz \\ \sum xy & \sum y^2 & \sum yz \\ \sum xz & \sum yz & \sum z^2 \end{bmatrix} = \begin{bmatrix} x_0 & & x_n \\ y_0 & \cdots & y_n \\ z_0 & & z_n \end{bmatrix} \begin{bmatrix} x_o & y_0 & z_0 \\ & \vdots & \\ x_n & y_n & z_n \end{bmatrix} \tag{4.1}$$

Singular value decomposition is performed on this covariance matrix which produces a set of three basis vectors $U = [\ \boldsymbol{\delta}_1 \quad \boldsymbol{\delta}_2 \quad \boldsymbol{\delta}_3\ ]$. Each basis vector is also an eigenvector of the covariance matrix whose magnitude is related to the plane fitting in each of the three Cartesian directions. So in the case of a flat plane, we expect the two largest eigenvectors $\boldsymbol{\delta}_1$ and $\boldsymbol{\delta}_2$ to represent the two axes parallel to the surface of the plane, as shown in Figure 4.2. The third and smallest eigenvector $\boldsymbol{\delta}_3 \approx \mathbf{0}$ indicates the axis perpendicular to the fitted plane; this is the desired normal vector of the plane. This is a space case where the ellipse defined by the basis vectors is flattened along one axis, and this effect is illustrated in Figure 4.2.




(a) (b)

Figure 4.2: Basis vectors of a covariance matrix for (a) the general case of some elliptical point distribution, and (b) the expected basis vectors for a set of points on a plane.

## 4.3 Ground Plane and Noise Reduction

When experimenting in uncontrolled environments it is likely that undesired data will be recorded. This extraneous data can come from multiple sources, including background noise, or superfluous features in a scene. The most prolific of these features is commonly the ground, so we attempt to identify and remove it from our data set prior to registration. This step is motivated by the fact that the ground plane may detrimentally affect the results of the ICP algorithm by introducing a large feature with inconsistent representation between images. As a result, we use a procedure to identify and remove the ground plane to avoid this potential problem.

We do this by determining the distribution of the $z$–coordinates (*i.e.*, height) of these points at a user defined resolution $r_{user}$, which we set to 1.5 cm. This resolution determines the number of bins the data occupies by way of Equation 4.2, rounding up

to the nearest integer. As a consequence of the rounding, the actual resolution $r_{rounded}$ needs to be determined algebraically using the reverse equation as shown in Equation 4.3.

$$n_{bins} = \left\lceil \frac{\max(\boldsymbol{z}) - \min(\boldsymbol{z})}{r_{user}} \right\rceil \tag{4.2}$$

$$r_{rounded} = \frac{\max(\boldsymbol{z}) - \min(\boldsymbol{z})}{n_{bins}} \tag{4.3}$$

The value of the resolution was selected in order to provide a small buffer in case the point cloud is slightly slanted, or the ground plane is not perfectly flat. In general, using a lower resolution (higher $r_{user}$ value) is better because selecting a wider buffer will reject more points, which may or may not be members of the ground plane. On the other hand, using a narrower buffer may cause points from the ground plane to span multiple bins.

In order for a bin to qualify as a candidate for the ground plane in our system, it must contain two additional standard deviations worth of points than the median. If $b_{ground}$ is the index of the "ground plane bin", then the criterion we use to identify it is summarized in Equation 4.4. We seek sharp impulses in the distribution that would indicate a large number of points concentrated at that height. Of the bins that meet the criterion, we declare the one at the lowest height $z$ to be the ground plane.

$$b_{ground} = \min(\boldsymbol{dist} > \text{median } \boldsymbol{dist} + 2\sigma_{dist}) \qquad b_{ground} \in \{1 \ldots n_{bins}\} \tag{4.4}$$

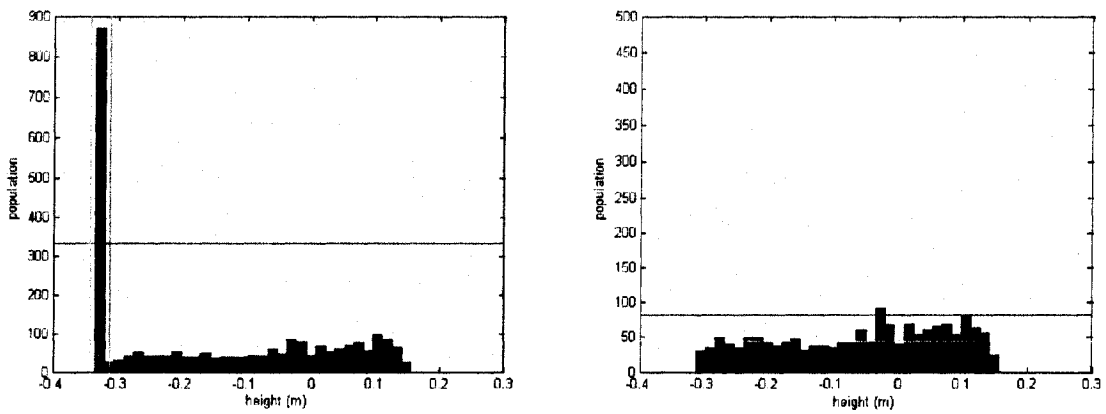
Point rejection is handled using the mean value of the range of heights spanned by $b_{ground}$ (*i.e.*, not the mean of the contents) as the center point of the rejection range. Subsequently, we remove all of the points in the range $2r_{rounded}$ centered at the middle of the "ground plane bin". This formulation is demonstrated in Equation 4.5. As mentioned previously, it may also be beneficial to choose a greater multiple of $r_{rounded}$ if it has a low value (*i.e.*, high resolution) to avoid the danger of missing outliers in adjacent bins.

$$\epsilon_1 = (\boldsymbol{z} \geq b_{Cground} - r_{rounded}) \cap (\boldsymbol{z} \leq b_{Cground} + r_{rounded}) \tag{4.5}$$

An example of a distribution used for ground plane identification is illustrated in Figure 4.3. The red line indicates the threshold used to identify possible candidates for $b_{ground}$ in this instance. We use an aggressive threshold to avoid false matches that may be caused by legitimate surfaces in the scene. In this case, the ground plane in Figure 4.3(a) includes a disproportionate percentage of the points in the scene, so this threshold is acceptable. In the case where it does not, retaining the ground plane is unlikely to interfere with alignment. In the case that we wish to scan a horizontal surface, the

distribution of the height is expected to produce a cutoff threshold that exceeds the population of any bin, thereby preventing accidental removal of horizontal surfaces.

Reviewing the figure again, the range between the green bars contains all of the points in $\epsilon_1$ that are rejected by our method, and the yellow bar indicates the median of the distribution. Figure 4.3(b) shows the same distribution after the ground plane has been removed, and the results appear excellent.




(a) (b)

Figure 4.3: Distributions of $z$–coordinates (a) before ground plane removal, and (b) after ground plane removal. The red bar indicates the threshold criterion for ground plane identification according to Equation 4.4. The yellow bar indicates the median value of the distribution, and the range between green bars indicates the point rejection range according to Equation 4.5.

Additional effort is made to remove outlying points that do not appear to be part of the object in the scene (*i.e.*, background noise). The process removes any points that are further than two standard deviations away from the median in both the $x$ and $y$ dimensions. This approach is implemented using Equation 4.6. It is a simple approach chosen for its speed to quickly reduce the background noise in our point clouds prior to alignment. The only disadvantage of this method is that it may occasionally crop useful information, but generally, this has no effect on the quality of our data registration.

$$\begin{aligned} \epsilon_2 &= (\boldsymbol{x} \geq \text{median}\boldsymbol{x} + 2\sigma_x) \cup (\boldsymbol{x} \leq \text{median}\boldsymbol{x} - 2\sigma_x) \\ \epsilon_3 &= (\boldsymbol{y} \geq \text{median}\boldsymbol{y} + 2\sigma_y) \cup (\boldsymbol{y} \leq \text{median}\boldsymbol{y} - 2\sigma_y) \end{aligned} \tag{4.6}$$

Then the complete set of points removed from each scene is summarized as follows:

$$\epsilon = \epsilon_1 \cup \epsilon_2 \cup \epsilon_3 \tag{4.7}$$

## 4.4 Integrating Sensor Motion

Since the INS dispenses attitude and heading information in real-time, it is a simple task to synchronize an image captured by the stereo camera with the absolute orientation of the sensor at the same moment. The same cannot be said for the translational motion of the sensor as it is moved from one viewpoint to the next because of the drift commonly associated with extended use of an accelerometer. Acceleration is sampled from the INS at up to 120 Hz and stored in a buffer which is emptied to file when a new frame of video is captured. Each sampled acceleration 3–vector is accompanied by a corresponding $3 \times 3$ orientation matrix because the capture rig can be moved with six degrees of freedom (DOF) at all times. This additional orientation information provides our system with the capacity to represent an acceleration vector in any coordinate frame. The translational motion between consecutive camera positions is integrated from acceleration data using a positional integrator.

In order for the integration to have any meaning, all of the acceleration values must be in a common coordinate frame, so we use the matrix transformations ${}^{w}_{s}R_{i=[1\ldots j]}{}^{s}_{c}R$ corresponding to each sample $j$ to put them all in a common frame (*i.e.*, the world frame in our case). We must also compensate for the acceleration caused by gravity which is known to be $G = [\ 0 \quad 0 \quad -9.82\ ]$ in the world frame. Once in a common coordinate frame, gravity is subtracted and the acceleration data is filtered using a $1^{st}$ order digital Butterworth filter with cutoff frequencies $f_L = 0.01\frac{f_s}{2}$ and $f_H = 0.9\frac{f_s}{2}$ Hz ($f_s = 120$ Hz) to minimize high frequency noise and low frequency drift.

Drift refers to low frequency cumulative error often due to temperature changes or other phenomena usually caused by prolonged use of the sensor and is a notoriously common problem with accelerometers. High frequency noise refers to impulsive noise that makes it seem as if the sensor was jerked in an abrupt manner and would be caused either by external contamination or just human error during data capture.

Figure 4.4 illustrates a set of typical $x$, $y$, and $z$ acceleration signals captured during the short interval of time between two consecutive camera positions. In general, the visible trend for the unfiltered signals is to slowly drift away from zero because of the various conditions that alter the accelerometer's steady state response over time. The

amount of drift in the filtered signals generally appears reduced, and each appears to have a steady state closer to zero than the raw version.

In order to combat drift, our system employs a Verlet integrator to compute translation from the acceleration data. Verlet integration is a finite difference method for numerically integrating the equations of motion. It has the benefit of being quite stable, especially in the face of enforced boundary conditions because there is no explicit velocity term with which the position term can lose synchronization [90]. Instead of storing position and velocity, this method stores current position and previous position. It is also very fast to compute and under the right conditions it is $4^{th}$ order accurate. By comparison, the Euler method is only $1^{st}$ order accurate, and the Runge-Kutta method is only $2^{nd}$ order accurate.

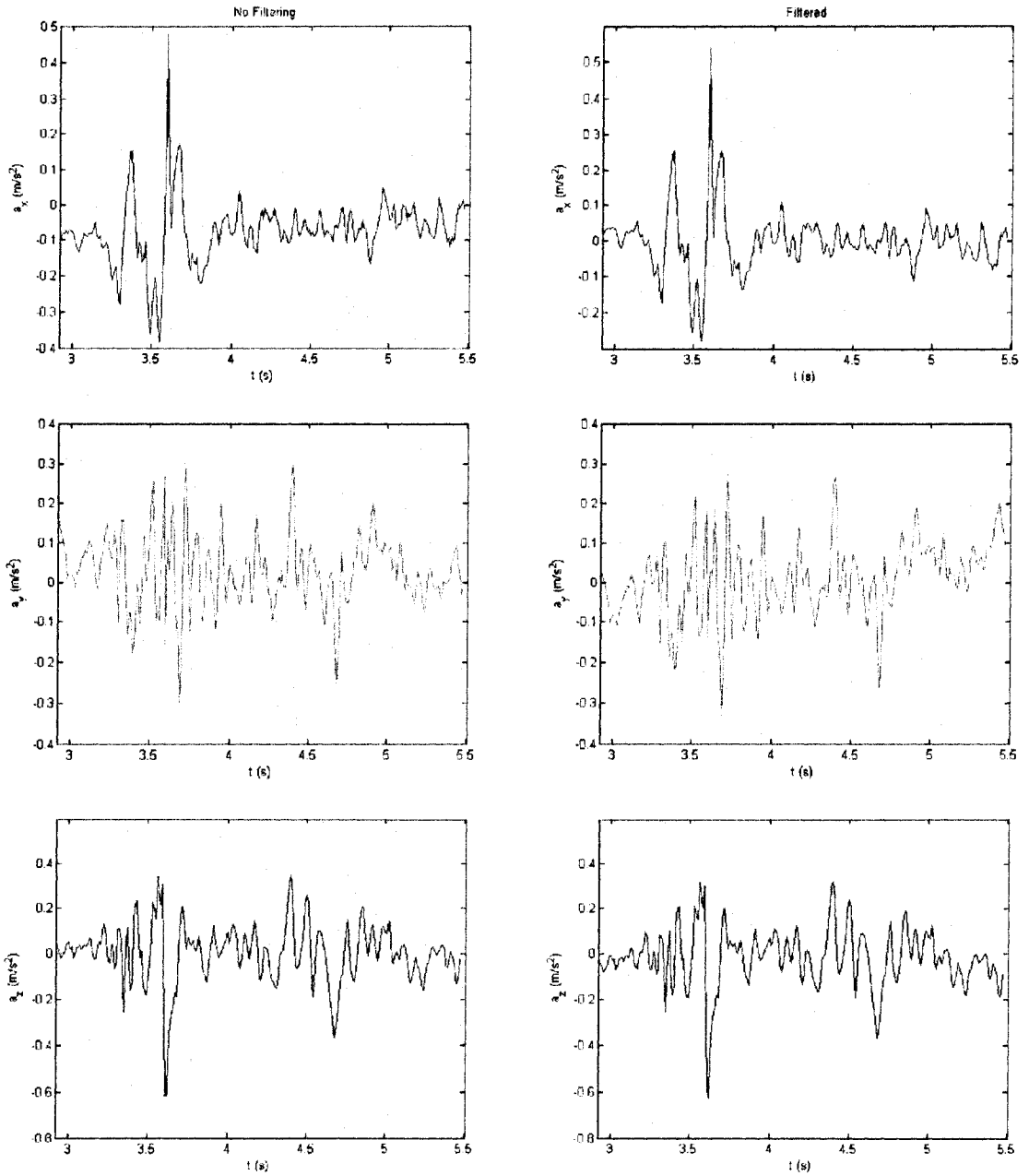
$$\begin{aligned} h &= t_i - t_{i-1} \\ x_1 &= -a_1(t_1 - t_2)^2 \\ x_{i+1} &= x_i + (x_i - x_{i-1}) + (h^2 a_i) \end{aligned} \tag{4.8}$$

The disadvantages of the Verlet method are that it handles changing time steps badly, and it is not a self–starter. That means it requires two steps to get going, so initial conditions are crucial. The standard Verlet function is shown in Equation 4.8, where $x$ represents integrated position data, $a$ the acceleration data, and $t$ the time vector for $i = \{1 \ldots j\}$ samples. We use it because it is a powerful method that maintains cheap numerical cost and high speed comparable to the Euler method.

## 4.5 Iterative Closest Point

The ultimate goal of a system like ours would be to use nothing but inertial navigation to align range information from different viewpoints in order to produce a complete model of a scene. Unfortunately, the hardware alone is currently insufficient because some of its tracking capabilities are still susceptible to measurement drift, and error induced by environmental or operational temperature changes. Aside from the performance limitations of the hardware, there remains a need for additional alignment between scenes because of error attributed to signal noise, inaccuracies in the stereo correspondence, or low resolution measurements.

Our system employs the iterative closest point algorithm for all of its multi–view data registration. The measurements from the navigation sensor are used as initial conditions (*i.e.*, a coarse registration) for the ICP algorithm in order to ensure a successful




(a) (b)

Figure 4.4: Typical acceleration signals recorded between two consecutive camera positions after compensating for gravity. The signals in (a) illustrate the raw $x$, $y$, and $z$ accelerations measured by the INS, and (b) illustrates the same signals after applying a first order band–pass filter.

convergence. Rusinkiewicz & Levoy [66] identified multiple variants of the algorithm, with multiple strategies for each of the six steps in their extended taxonomy. The ideal variant for any particular application generally depends on the properties of the data set, but a few of the techniques that were reviewed appear to be globally beneficial. Our system is currently limited to techniques that work on a point to point basis because all of our data is acquired and stored as point clouds. Generating a full set of surfaces for each individual image would also be highly impractical, especially in large data sets. Additionally, there is no guarantee that the reconstructed surfaces would be entirely correct, and may inadvertently contribute error to the registration.

The task of selecting a set of points for matching in our system is performed using normal-space sampling [66]. This procedure is accomplished by converting the normal vector of each vertex from Cartesian coordinates to spherical coordinates by way of equation Equation 4.9, and bucketing points according to the position of their normal in spherical space, then sampling as uniformly as possible across the buckets. In our implementation, the buckets measure $5^\circ \times 5^\circ$ because a smaller size typically results in an excess of empty or poorly populated buckets, while a larger size typically does not discriminate well enough for this technique to work. Some statistics relating bucket size to bucket population are provided in Table 4.2.

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \arctan(\tfrac{y}{x}) \\ \phi &= \arccos(\tfrac{z}{\rho}) \end{aligned} \tag{4.9}$$

| **Bucket Size** | **Total # of Buckets** | **Average # of Empty Buckets** | **% of Empty Buckets** | **Average # Per Bucket** | **Average $\sigma$ of # Per Bucket** |
|---|---|---|---|---|---|
| $1^\circ \times 1^\circ$ | 32400 | 5633 | 17.39% | 7.91 | 9.12 |
| $5^\circ \times 5^\circ$ | 1296 | 30.50 | 2.35% | 151.03 | 177.87 |
| $10^\circ \times 10^\circ$ | 324 | 1.78 | 0.55% | 593.01 | 697.30 |
| $15^\circ \times 15^\circ$ | 144 | 0.16 | 0.11% | 1328.40 | 1530.00 |
| $20^\circ \times 20^\circ$ | 90 | 0 | 0% | 2122.90 | 2631.30 |

Table 4.2: Normal space sampling statistics relating bucket size to bucket population over one hemisphere for our primary data set, which is described in Section 5.1.

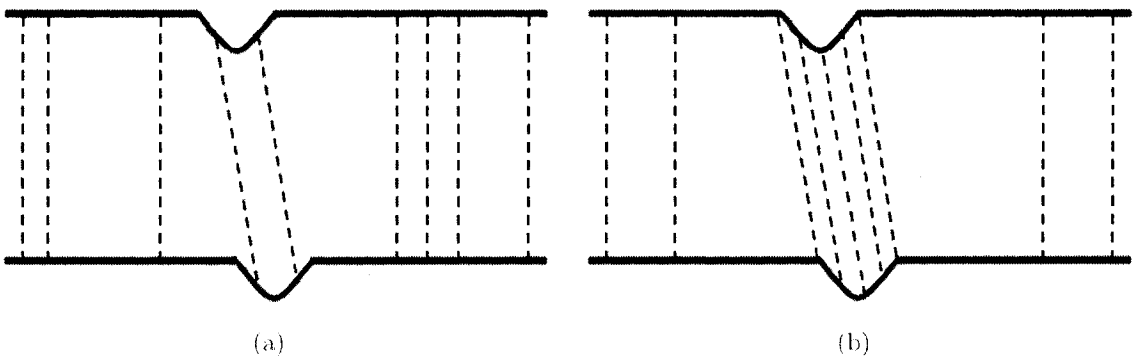



Figure 4.5: Plausible point pairs resulting from (a) "random sampling", and (b) "normal–space sampling" strategies. Using random sampling, the sparse features may be overwhelmed by presence of noise or distortion, causing the ICP algorithm to not converge to a correct alignment. Figure adapted from [66].

Normal–space sampling ensures that small features (*i.e.*, features composed of fewer points) in a scene that may be vital to convergence receive equal representation in the sampled set as other, more prominent ones. It significantly outperforms uniform and random sampling in these cases, and even demonstrates marginal improvement applied to other data sets. This effect is demonstrated in Figure 4.5, which highlights the points that would be sampled using normal–space sampling versus random sampling for sparse features. The disadvantage of using this method is that we need *a priori* knowledge of the vertex normals for all of the points acquired by the camera, which is why they are estimated prior to registration using the method described earlier in this chapter.

Point matching in our system is performed using another normal–based method that seeks to match points from one set to the "closest compatible point" in another based on the angular alignment between their normals. This technique is demonstrated in Figure 4.6, and illustrates the expected point correspondences using two different strategies. While not infallible, the closest compatible point strategy is a marked improvement over the basic closest point strategy.

Rusinkiewicz & Levoy report that this matching criterion performs better than the standard unbiased point–to–point technique, but normal–shooting and reverse projection perform much better than both in all cases, except when aligning diminutive features [66]. Our choice was primarily motivated by the lack of surface meshes required to use either of these techniques, and as we mentioned before, constructing them would be impractical because our system uses sparse, subsampled data to speed up registration.

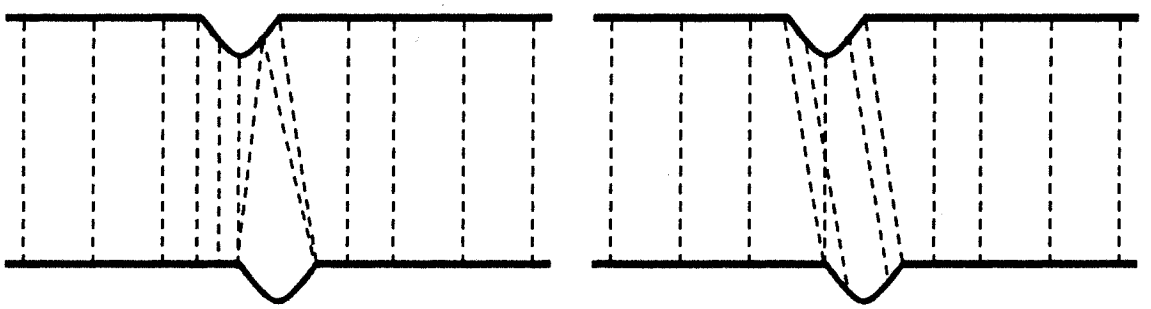



Figure 4.6: Expected point correspondences using (a) the "closest point", and (b) the "closest compatible point" strategies. Using closest point matching, many valid feature points may be entirely overlooked during matching if other closer alternatives exist.

Computing the angular distance between two normals is a trivial matter. Equation 4.10 is used to validate whether a pair of points is a compatible match.

$$2\text{Re}\{\arccos(|\boldsymbol{n}_1 \cdot \boldsymbol{n}_2|)\} < \theta_c \tag{4.10}$$

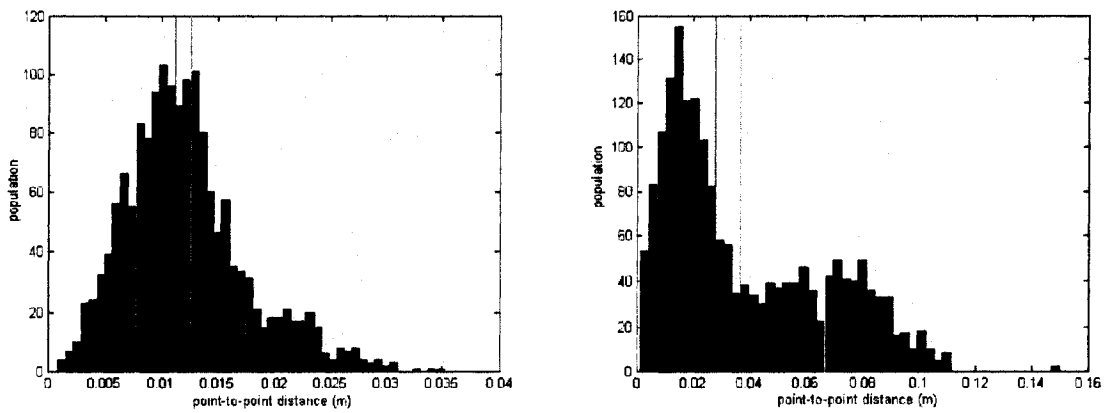
Based on the compatibility data summarized in Table 4.3, we selected a cutoff threshold $\theta_c$ of 30°.

| **Angle** | **Average # of Compatible Points** | **Average % of Compatible Points** |
|---|---|---|
| 5° | 8.73 | 0.29% |
| 15° | 71.69 | 2.23% |
| 30° | 262.46 | 8.09% |
| 45° | 524.64 | 16.19% |
| 60° | 833.85 | 25.61% |

Table 4.3: Statistics indicating the number of compatible points found relative to the cutoff angle employed when using the "closest compatible point" point matching strategy on our primary data set, which is described in Section 5.1.

Our system does not currently utilize any point weighting scheme, but it does include a point rejection step. After a set of compatible points have been identified, we determine

the distribution of the point–to–point distances of that set. Some sample distributions are shown in Figure 4.7. Any points further apart than some multiple of the median distance reported by the distribution are discarded. We selected the cutoff distance to be twice the value of the median because it is less susceptible to outliers than standard deviation, or mean. This property implies that in most cases the median of our point–to–point distance distributions will be lower than the mean as well. We exploit the stability of the median to make our rejection rate more consistent, particularly in cases where the distribution appears similar to Figure 4.7(b), and deviates strongly from from the ideal curve.




(a) (b)

Figure 4.7: Distributions of point–to–point distances for two different applications of our ICP method that denote (a) a pair of point clouds nearing alignment that requires few rejections, and (b) a partially aligned pair of point clouds demanding significant rejection. The red lines indicates the median in each case, the green lines indicates the mean, and the yellow lines indicates one standard deviation from the mean.

Since the purpose of point rejection is to eliminate outliers, the impact of this step is not as large as others. Most of the alternative rejection methods reviewed in Chapter 2 yield similar results, and they do not show much improvement over simply retaining all of the points. The reason for this is likely because outliers generally make up a very small part of the data set in the first place. The advantage of our method is that it determines a per–view optimal distance measure dynamically, while a static measure would not apply equally well to all views. Additionally, the multiplier constraint can potentially be modified to control the rate of rejection to keep only the closest points, or

to limit the number of discarded pairs.

Referring to Figure 4.7, we can visually approximate the number of points that would be discarded using a median, mean or standard deviation distance measures. It is apparent that the median and the mean are the most stable between the two distributions, which are clearly different. An approach utilizing standard deviation appears to produce more varied rates of rejection for any given set of points.

Lastly, the error metric used by our system is the standard sum of squared distances, minimized over a set of points generated using the current transformation. We set the point–to–point distance criterion (*i.e.,* the ICP error threshold) for each pair of point clouds to be 3 mm, as indicated in Table 4.1. It is an easy technique to implement, but our use of this method was once more motivated by the point–based representation of our data. Generally speaking, point–to–point error metrics do not converge as reliably as well as point–to–plane techniques [66], but within each category the differences in performance are minimal.

## 4.6 Heuristics

Our system uses a scheme that compares the angular distance between the orientations of the camera in adjacent frames to group similar sets of points together in an effort to speed up and improve the quality of alignment between them. The maximum number of members allowed in each of these "similarity groupings" is determined initially by finding all of the consecutive frames that fall within a certain angular threshold of the first frame in the data sequence. However, each group is required to possess at least two members, so the smallest possible group is two consecutive frames. For example, a complete data sequence contains 50 frames in total, and frames 2 to $n$ ($n \leq 50$) fall within some predefined angular threshold distance of frame 1, then no subsequent group can exceed $n$ members. Each subsequent group is automatically constructed by choosing a different set of $n$ consecutive frames, and rejecting those that do not meet the angular distance criterion when compared against the first frame in their group.

We set the angular threshold distance in our system to 30°; relative distances between the members of each group are computed using Equation 4.11 where the camera orientations are represented by quaternions.

$$q = [\ r_0 \quad r_x \quad r_y \quad r_z\ ]^T$$

$$\Delta_{1 \leftarrow i} = 2\mathrm{Re}(\arccos(|q_1 \cdot q_i|)) \qquad i \in \{2 \ldots n\} \tag{4.11}$$

The total number of groups can vary depending on how much overlap is desired between adjacent groups. The motivation for this grouping scheme in our system is two–fold:

1. Identifying frames that portray the same part of a scene makes it easier to quickly and reliably align entire sections of the reconstruction. Similarly, it prevents any attempts to match views that do not share any common elements, and would ultimately end in failure.

2. When all of the members of a group are aligned, the redundancy of the larger set of points produced by grouping can potentially benefit inter–group registration by increasing the number corresponding point pairs. Effectively, this counters some of the information loss caused by the heavy subsampling of the raw data using the method described earlier.

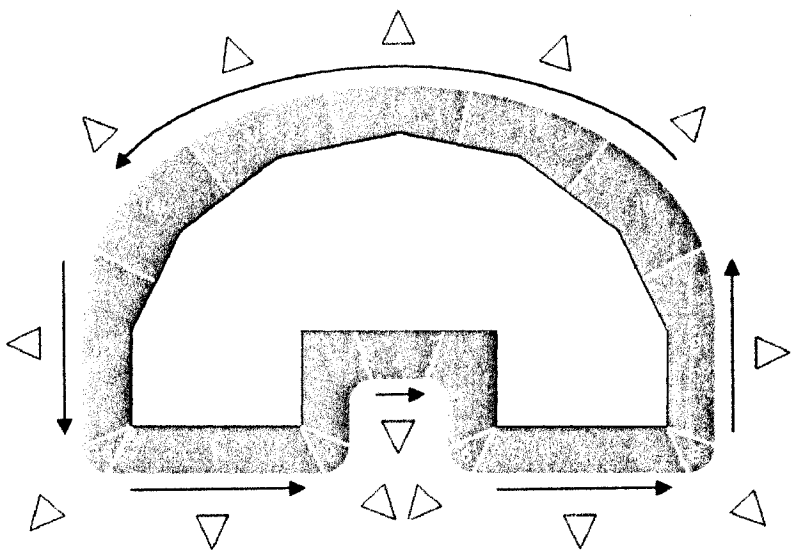


Figure 4.8: Theoretical "similarity groups" for the indicated camera motion, assuming a constant distance from the camera to the scene. Each numbered section indicates where the camera orientation falls within an angular distance threshold of roughly 30°. The number of individual frames contained in each section depends on the rate of motion of the camera, and the number of groups in each depends on how much overlap is desired between groups.

With this in mind, we opt to use the most conservative approach for this task which retains the maximum amount of overlap between groups. This means that each frame in

the sequence, except the last, will act as the reference frame of its own group (*i.e.*, will be compared against). Alternatively, a more aggressive approach could be used where the overlap between groups is not as great in order to reduce the amount of redundant information produced by large amounts of overlap, and to increase the overall speed of the system. This implies that there would be fewer groups, and only some subset of frames would act in a reference frame capacity. This option merits further exploration, but due to the fact that group sizes can vary greatly based on the motion of the camera, it is likely that many of the group arrangements will degenerate into the first scenario. These two simple arrangements are illustrated in Figure 4.9.

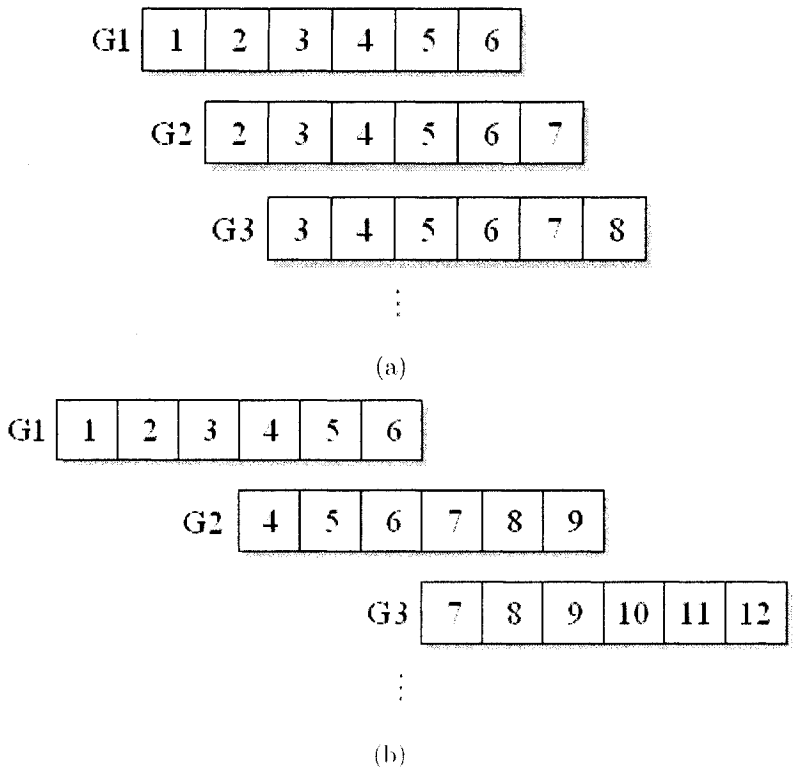



Figure 4.9: Examples of the first three "similarity groups" of a longer sequence constructed (a) with maximum overlap between groups, versus (b) less overlap between groups. This example assumes the maximum number of members for each of the groups following the first, but this is not generally the case. Extrapolating further, it is apparent that while every frame defines its own group in the first case, only a subset of all the frames does so in the second case.

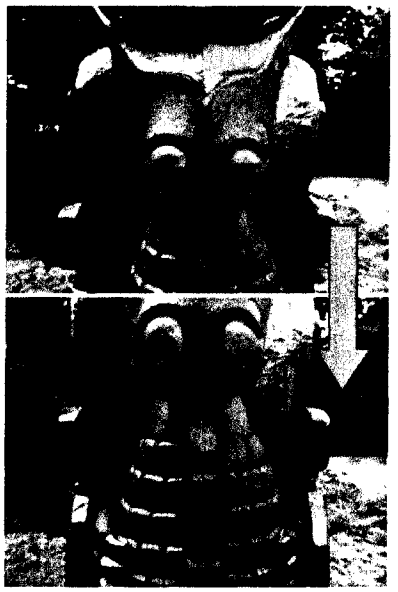
While the inertial navigation sensor provides good initial conditions, they are not perfect. As a result, our system uses the iterative closest point method we described earlier in Section 4.5 to improve upon the measurements taken from the INS. However, ICP can be fickle and extra care must be taken to select a procedure that is robust enough to handle any generic sequence of data.

In general, there are two extremes for the types of data sequences our system should encounter. The first involves primarily translation between frames such as the scenario where the camera is moving parallel to a wall or relief. The second type involves primarily rotation which would occur when capturing something like a statue. Both types are illustrated in Figure 4.10. The properties of the data sequence itself may result in either false positive or false negative point cloud registration. Consequently, different data sequences influence the outcome of the registration in different ways, and may require additional processing or error checking to produce the correct global solution.

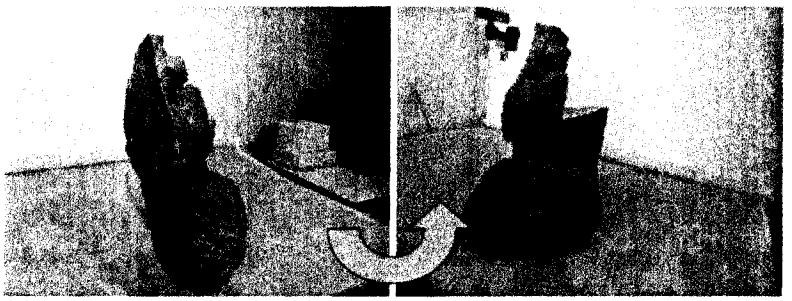
With this in mind, our system uses sequential registration to match point clouds, whereby each set of points is matched against an immediate neighbour. An alternative to our method involves registering all views against a single reference view. The flow of the two methods is illustrated in Figure 4.11. Although the sequential method takes longer to execute than the alternative, it actually helps to further combat the error present in the acceleration signal. The reasoning is that when two neighbouring views are registered using this strategy, the error between them is minimized. This is a prudent measure because the major disadvantage in using ICP is a failure due to poor initial conditions (*i.e.*, sensor data, integrated position).

For example, assuming a sequence of four views, such as the one in Figure 4.11(a), the total error observed when aligning the first and last views using sequential matching is $\epsilon_s = \frac{1}{2}\,\epsilon + \frac{2}{3}\,\epsilon + \frac{3}{4}\,\epsilon$. Similarly, the total error observed when aligning the first and last views directly is $\epsilon_r = \frac{1}{4}\,\epsilon$. In general, $\epsilon_s \leq \epsilon_r$ because of the distance and dissimilarities between these two views, which implies better performance using sequential matching. Furthermore, the use of a common reference frame completely ignores the relationships between neighbouring point clouds, and is therefore unlikely to converge to the correct global solution because the error between adjacent views will not have been minimized.

Specifically, our system performs a three–stage registration on the "similarity groups" which were described in detail earlier in this section. The first stage refers to *intra group registration*, whereby the contents of each group are registered against other members of the same group. During the second stage, our system performs *inter group registration*, which aligns all of the completed groups relative to each other. The last stage includes a



(a)

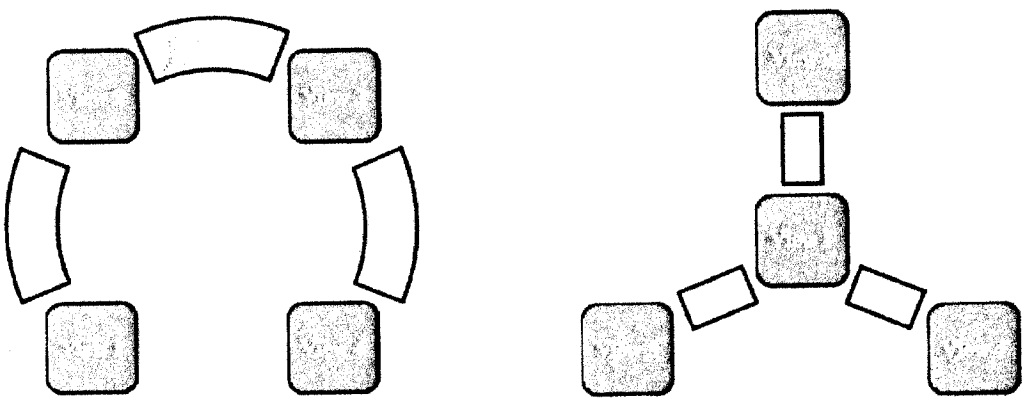


(b)

Figure 4.10: Example of (a) a primarily translational data sequence, and (b) a primarily rotational data sequence. Our use of a hand–held setup implies that it is nearly impossible to ever achieve a pure motion of either type.

final *global registration* stage that is designed to remove the small alignment errors that remain between the groups. Such errors often accumulate and produce misalignments at the end of a sequence when working with a single object (*e.g.*, statue or building) that is sampled on a closed loop.

During registration, the system generally works with data that has been heavily downsampled using the normal–sampling method discussed in Section 4.5. The removal of such a large number of points may increase the difficulty with which the ICP algorithm is able to locate good point–to–point correspondences. In order to reduce the impact of



(a) (b)

Figure 4.11: Flow–chart for (a) sequential registration of point sets, and (b) registration of all point sets against a single reference.

heavy downsampling, our system registers each member of a group against the combined set of all of the members that come before it. Every time the ICP algorithm is used to align two sets of data points, the translation and rotation that produce the best results are stored in a $4 \times 4$ homogeneous transformation matrix ${}^{q}_{z}T$, according to the notation discussed in Chapter 3. This extensive bookkeeping facilitates the alignment of previous group members, to more recent ones in the sequential registration. The goal is to provide the ICP algorithm with a larger set of reference points from which to choose correspondences for the current point cloud in the sequence.

Consequently, assuming a sample group containing $s \geq 2$ individual sets of points, then *intra–group registration* can be executed using Equation 4.12 where $\boldsymbol{P}_k$ is the $k^{th}$ set of points, and $k \in [2 \ldots s]$. The registration between the first two members of a group is a straight forward one–to–one comparison. However, subsequent registrations first require all other prior group members to be represented in the coordinate frame of the immediate predecessor to the element that is currently under consideration (*e.g.*, $\boldsymbol{P}_3$ would be registered against ${}^{2}_{1}T\boldsymbol{P}_1 \cup \boldsymbol{P}_2$, and $\boldsymbol{P}_4$ against ${}^{3}_{2}T\ {}^{2}_{1}T\boldsymbol{P}_1 \cup\ {}^{3}_{2}T\boldsymbol{P}_2 \cup \boldsymbol{P}_3$, *etc.*).

$$
{}^{k-1}_{k}T \;=\; \begin{cases} \text{ICP}([\ \boldsymbol{P}_{k-1}\ \ \boldsymbol{P}_k\ ]), & k = 2 \\ \text{ICP}([\ \boldsymbol{P}_{k-1} \cup_{m=1}^{k-2} (({\textstyle\prod_{i=1}^{m}}\ {}^{k-i}_{k-i-1}T)\boldsymbol{P}_{k-m-1})\ \ \boldsymbol{P}_k\ ]), & 3 \leq k \leq s \end{cases} \tag{4.12}
$$

Once this procedure has been applied to all of the point clouds in one group, the compound transformation shown in Equation 4.13 is used to represent them all in the reference coordinate frame for that group (*i.e.*, coordinate frame of the first point set)

prior to being merged into a single large point cloud which will be utilized during *inter-group registration.*

$$P_m = (\prod_{i=1}^{m-1} {}^{i}_{i+1}T)P_m, \qquad m \in \{2 \dots s\} \tag{4.13}$$

On the other hand, *inter-group registration* is a purely sequential method where each pair of groups is compared in a one-to-one fashion with no additional steps because a group consists of multiple smaller point clouds by definition. In this case, there should already be a sufficiently large selection of points for the ICP algorithm to locate good correspondences. Inter-group registration can be summarized for $s \geq 2$ groups by Equation 4.14 where $\boldsymbol{P}_k$ is the $k^{th}$ group, and $k \in [2 \dots s]$.

$${}^{k-1}_{k}T = \text{ICP}([\ \boldsymbol{P}_{k-1} \quad \boldsymbol{P}_k\ ]) \tag{4.14}$$

When all of the groups have been aligned in this way, Equation 4.13 is employed once more to transform them each into the coordinate frame of the first group, which acts as the reference for this stage of the registration. This produces a nearly finished set of vertices that defines the surface of the subject photographed during the stereo video capture. Unlike the previous stage where all of the point sets in a group were merged into a single entity, the groups remain separate entities to facilitate the third stage of registration. The *global registration* phase is executed using third-party software like Geomagic Studio[29] or MeshLab [50] which also apply some variant of ICP to eliminate the small errors that remain between groups, and to close the loop between the last and the first group of a closed-loop capture sequence (if applicable).

As mentioned earlier, ICP can be fickle when working with different data and additional processing or error checking is needed to generate the correct alignments during the *intra-* and *inter-group registration* stages. The orientations provided by the sensor are drift free (according to manufacturer specifications) and assumed to be correct, so the system utilizes the ICP algorithm discussed in Section 4.5 both with and without the rotational component in its heuristics. Ignoring the estimate for rotation at each iteration results in the alignment of the mean of the correspondence pairs, which may be enough to produce the best result if the orientations of the data are already correct. Separating translation from rotation is more likely to correct the remaining inaccuracies in the relative positions of two point clouds without introducing a new rotational error in the process; such errors could arise if there are enough bad correspondences.

A rudimentary state machine is used to execute multiple trial and error procedures to determine the best result that falls within the error threshold specified by the user

during initialization. The order of these procedures is selected in a way that maximizes processing speed while maintaining generality when working with the different varieties of data sequences discussed earlier. For example, good initial conditions produce fast and easy ICP matches, so the data is first subjected to either translational or standard ICP. If this trial should fail to produce results within desired parameters after a certain number of tries (*i.e.*, the user defined escape condition introduced in Section 4.1), the task is handed down the line for additional processing. Subsequent processes are executed in an increasingly time intensive order so that faster trials are executed sooner, rather than later.

Our system utilizes the following trials to determine the optimum alignment between every pair of point clouds $\boldsymbol{P}_r$ (*i.e.*, reference points) and $\boldsymbol{P}_c$ (*i.e.*, points to be aligned) in the data sequence that meets the user specified distance error measure:

1. Register the two point clouds $\boldsymbol{P}_r$ and $\boldsymbol{P}_c$ using ICP ignoring the rotational component at each iteration (*i.e.*, translational ICP) until a good result is found, or the escape condition is met. If no match is found the task is handed down the line to the next process.

2. Register the two point clouds $\boldsymbol{P}_r$ and $\boldsymbol{P}_c$ using full ICP until a good result is found, or the escape condition is met. If no match is found the task passes to the next process.

3. Align the median of current point cloud $\boldsymbol{P}_c$ with the median of the "last view" prior to applying translational ICP, then continue until a good result is found, or the escape condition is met. If no match is found the task is handed down the line to the next process.

4. Align the median of current point cloud $\boldsymbol{P}_c$ with the median of the "last view" prior to applying full ICP, then continue until a good result is found, or the escape condition is met. If no match is found the task passes to the next process.

5. Align the medians of the two full point clouds $\boldsymbol{P}_r$ and $\boldsymbol{P}_c$ before applying translational ICP, then continue until a good result is found, or the escape condition is met. If no match is found the task is handed down the line to the next process.

6. Align the medians of the two full point clouds $\boldsymbol{P}_r$ and $\boldsymbol{P}_c$ before applying full ICP, then continue until a good result is found, or the escape condition is met. If no match is found the task passes to the next process.

7. If no result has been found that meets the defined point–to–point distance error criterion, then any changes made to the points $\boldsymbol{P}_c$ in previous steps are discarded and additional normal–sampling is performed on the point set $\boldsymbol{P}_c$ currently under review. Samples from any bins containing more than the median number of elements observed across all of the bins are removed, and the system repeats steps 1–6. This step is executed only once for each pair of $\boldsymbol{P}_r$ and $\boldsymbol{P}_c$.

8. If the heuristics fail to locate an acceptable solution, the points $\boldsymbol{P}_c$ are discarded.

The definition of the "last view" during the *intra–group registration* stage is different from that of the *inter–group registration* stage. In the former, it refers to the immediate predecessor of the point cloud currently being registered (*i.e.,* some subset of $\boldsymbol{P}_r$), as opposed to the union of all of the preceding point sets in the group. In the latter, it simply refers to the points that comprise the previous group in the sequence (*i.e.,* the entire set of $\boldsymbol{P}_r$). Figure 4.12 illustrates the flow of information through our heuristics.

Resultant vertices and their corresponding per–vertex colour information are stored using the PLY (*i.e.,* Stanford Polygon) file format for each individual group in order to facilitate further post–processing (*e.g.,* surface reconstruction, noise reduction, texturing, quality assurance, *etc.*) using third–party applications. Additionally, we put forth the effort to store a copy of the individual reference frames from each group separately for quality assurance purposes.

## 4.7 Surface Reconstruction

The product of data registration is a dense cloud of vertices in the shape of the surface of the subject that was photographed during stereo video capture. As mentioned before, following registration and alignment, the point clouds that comprise a data sequence are exported to Stanford University PLY files (PoLYgon). This particular format was chosen because it supports pure vertex data like the Wavefront OBJ format, but it also supports per–vertex colour information which can be easily extracted from captured video. It is a simple file format with extensible user–defined properties that may be utilized in future work.

Immediately after registration, the resultant model is typically a rough estimation of the surface shape of the subject with many superfluous vertices, and background noise from the environment. Our surface reconstruction and other post–processing is done by importing the PLY files exported from Matlab into third–party software such as MeshLab

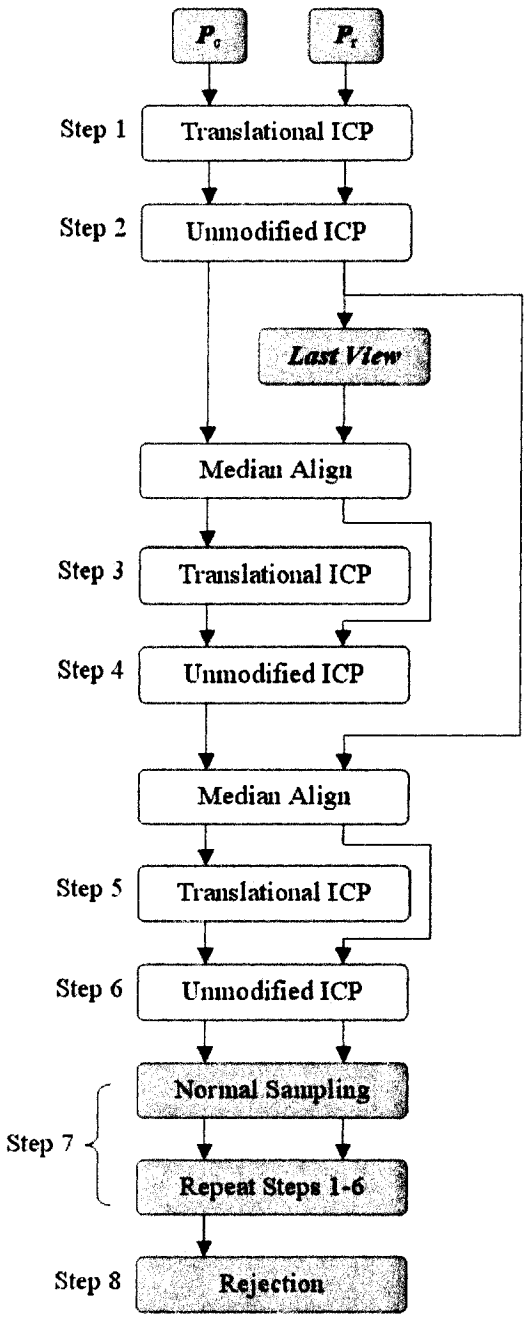
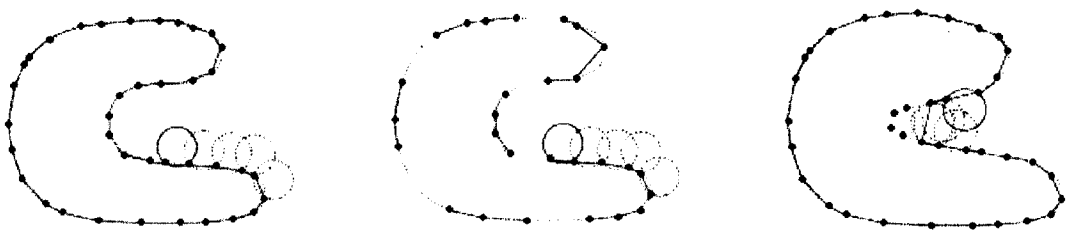



Figure 4.12: The flow of information through the various steps of our heuristic registration.

[50] or Geomagic Studio [29]. MeshLab is an open source, portable, and extensible system

for the processing and editing of unstructured 3D triangular meshes. It is aimed to help the processing of "the typical not–so–small unstructured models arising in 3D scanning, providing a set of tools for editing, cleaning, healing, inspecting, rendering and converting that type of data [50]". Geomagic Studio is "an advanced piece of commercial software designed to digitally reconstruct real world shapes from 3D scan data and create accurate models." It is aimed to accelerate product design, re–engineering, mass customization, engineering analysis, rapid prototyping and digital archiving [29].

There are several methods with which to build a surface mesh from a raw point cloud (*e.g.,* some popular Delaunay based methods [92, 64]), but the one employed by our system is the ball–pivoting surface reconstruction algorithm [7] available in Meshlab. Three points form a triangle if a ball of a user–specified radius touches them all without containing any other point in the cloud. The algorithm begins with a single seed triangle where a ball sits. The ball pivots around an edge without losing contact with the points on that edge until it touches another point thereby forming another triangle. The process continues until all the points in the cloud have been considered [7]. The two–dimensional case of this algorithm and some of its limitations are summarized in Figure 4.13. In our procedure, we allow MeshLab to analyze the point cloud to determine the optimal radius for the ball used in the algorithm.

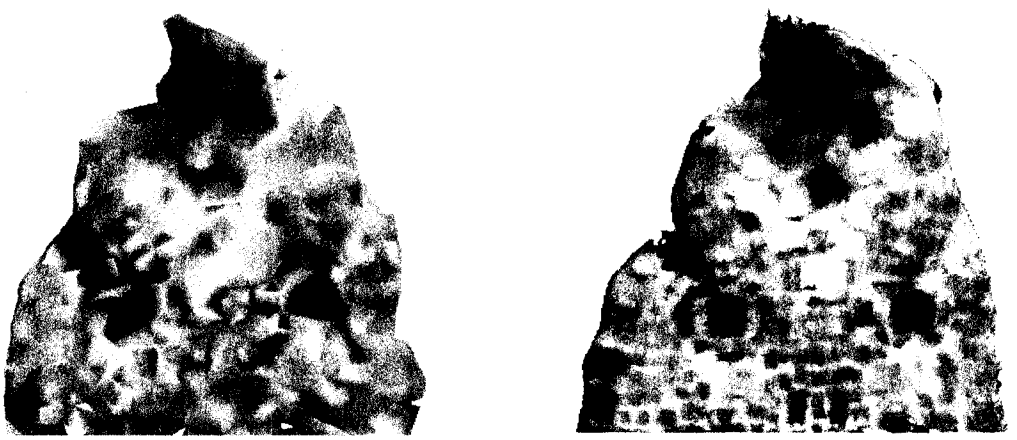


(a) (b) (c)

Figure 4.13: The two–dimensional ball–pivoting algorithm where: (a) a circle of radius $\rho$ pivots correctly from one sample point to another and connects them with edges, (b) an excessively low sampling density causes holes in the surface because some of the edges cannot be created, or (c) the curvature of the manifold is larger than $\frac{1}{\rho}$, resulting in missing features because some sample points will not be reached by the pivoting ball. Figure adapted from [7].

At this point we have a basic surface model to work with, but it is still rough and untextured. The next step in the 3D modeling pipeline aims to provide a texture to any

object captured by our scanning system. Texturing 3D models that are created manually for complex applications like video games also have custom texture maps which are nearly as much work to create as the 3D surface on which they are mapped. Fortunately, our system has access to the photographs from which the 3D model was constructed. Our system produces textures by simply interpolating surface colours from vertex colour information extracted from video. We use interpolation instead of generating a texture map because it functions equally well across all resolutions with varying texture quality. More vertices (*i.e.,* a high sampling density) will mean higher texture resolution and more accurate and realistic results, while lower resolutions (*i.e.,* a low sampling density) will produce blurrier textures because of the interpolation process. This effect is demonstrated in Figure 4.14. Conversely, generating a texture map from video may not necessarily ignore outliers like interpolation, and it entails repeated mapping of texture coordinates for each desired resolution.



(a) (b)

Figure 4.14: Samples of the textures produced by interpolation when employing (a) a low sampling density, and (b) a high sampling density.

Typically, it is desirable to execute the stereo registration procedure using heavily downsampled data because the process would be prohibitively slow otherwise. Clearly, this is counter productive when trying to generate high resolution models and textures. The simplest and most effective solution to this problem is to first execute the data registration using heavily subsampled data, then apply the resulting homogeneous transformations to the full quality original data set. This way the system can achieve high speed registration without any loss in the surface accuracy of the model, or texture

quality.

Additionally, it may be prudent to clean the model, which involves removing faces and vertices that are superfluous or appear incorrect. These include unreferenced vertices, duplicate vertices, zero area faces, non-manifold faces, or vertices of a certain colour. Some of our preprocessing tasks outlined in Section 4.3 include noise reduction, but the purpose of those tasks is much more specific than the types listed here. Isolated pieces of the mesh can be automatically removed based on certain criteria: either the number of faces in the piece, or its distance from the origin of the mesh. Lastly, any holes that should not exist in the surface mesh can be sealed. All of these tasks can be performed easily using MeshLab filters. Additionally, a smoothing function like a Laplacian, HC Laplacian [91] or TwoStep would reduce the noise in the model, but may be undesirable because it could result in a loss of detail on jagged surfaces. However, the use of these techniques is highly subjective, and depends entirely on the needs of the operator.

## 4.8 Summary

In this chapter, we discussed the various techniques and procedures utilized by our 3D scanning system to condition and align stereo data points from multiple camera viewpoints. In Section 4.1 we outlined the steps and settings utilized by our system to prepare for subsequent processing, and Section 4.2 describes the technique employed to estimate the vertex normals for each point cloud. We also attempt to condition the data from both sensors with filtering and digital processing to reduce background noise by removing the ground plane any outlying points from the stereo data, and to reduce drift and high frequency noise in the acceleration data. These processes are described in detail in Section 4.3 and Section 4.4, respectively.

In Section 4.5, we identify the ICP variant utilized for data registration, and Section 4.6 outlines the processing tasks used to determine the optimum alignment between every pair of point clouds in the data sequence that meets the desired distance error measure. It also described our "similarity groupings" which lump together neighbouring point clouds based on the orientation of the camera to facilitate faster convergence during ICP. Our system employs a three-stage registration process that includes *intra group registration*, *inter group registration*, and *global registration*.

- *Intra group registration* refers to the alignment of the contents of each group using the heuristic processing tasks discussed earlier with our ICP variant.

- *Inter–group registration* refers to the alignment of all the groups using our ICP variant with the heuristic processing tasks itemized earlier in this chapter.
- *Global registration* refers to the minimization of any remaining error between groups, and closing the loop between the last and the first group of a closed–loop capture sequence.

Lastly, Section 4.7 details the techniques used to reconstruct a textured surface for the frame of vertices produced by the data registration stage of our system. While not explicitly required, we also detail some additional post–processing steps that the end–user may want to perform in order to produce the highest quality model possible.

# Chapter 5

# Experimental Results

In this chapter we present practical results produced when using our 3D scanning system. Section 5.1 profiles our primary data set which was modeled fully using our automated system. In Section 5.2 we discuss the effectiveness of our three–stage registration procedure, highlighting the intra– and inter–group alignment errors corrected by our system to produce a rough final model prior to global registration. While internal error measurements are a good gauge of the local alignment in a small neighbourhood of views or groups, they provide no information about the global correctness of the final model. Consequently, in Section 5.3 we compare the final model constructed by our system directly to the "ground truth" source material.

Furthermore, since the performance of a 3D registration algorithm often depends on the characteristics of the data, we execute due diligence by selecting an alternate data set with vastly different characteristics which is profiled in Section 5.4. In the process of testing, some limitations were discovered in our system which we outline in Section 5.5. Lastly, we summarize our findings in Section 5.6.

## 5.1 Primary Data Profile

Our primary data set is a closed–loop sequence 50 frames in length consisting of an abstract statue of an owl placed on a concrete path just outside the Colonel By building at the University of Ottawa. We selected this statue to be our first subject because its many varying surface qualities facilitate good stereo correspondence and registration; it includes a bumpy convex surface, a smooth tiled surface, a flat surface with deep impressions, and complex solid geometry. This data set will henceforth be referred to as

the "artwork" sequence; a few sample frames are illustrated in Figure 5.1 and the full 50 frame sequence is available in Appendix C.1.

(a) (b)

Figure 5.1: Subject of the primary "artwork" data set as it appears (a) from the front, and (b) from the rear.

The artwork stereo sequence was captured at a resolution of $1024 \times 768$ pixels using the same settings as those that were selected for our calibration routine in Section 3.1. The data was subsampled by a factor of 100 for fast registration, and by a factor of 5 for surface reconstruction and texturing in order to improve the appearance of the final product. The calibration matrix for the hardware configuration used to acquire this data was determined by our routine to be:

$$ {}^{c}_{s}R = \begin{bmatrix} -0.0407 & -0.0330 & 0.9986 \\ -0.0120 & 0.9994 & 0.0326 \\ -0.9991 & -0.0107 & -0.0411 \end{bmatrix} \tag{5.1} $$

Additionally, it was acquired using a smooth clockwise motion in a single revolution of the subject with the camera held about half a meter to a meter away, as necessitated by the disparity range presented in Table 3.1. This type of acquisition motion can be classified as mostly rotational due to the constant changes in the orientation of the camera, and it is one of the types that we discussed in Section 4.6. Most of the testing and debugging for our system was done using this data set.

Lastly, in Table 5.1 we present the number of times each step in our heuristics successfully facilitates convergence that meets the user defined error criterion for this data set.

These statistics are a reflection of the quality of the data and the coarse registration in addition to the performance of our heuristics because good initial data does not require much conditioning, or many iterations to reach the global solution.

| Step | # of Alignments |
|---|---|
| 1 | 5 |
| 2 | 122 |
| 3 | 12 |
| 4 | 10 |
| 5 | 17 |
| 6 | 6 |
| 7 | 18 |
| 8 | 0 |

Table 5.1: Statistics indicating the number of successful convergences that occurred during each step of our heuristics as outlined in Section 4.6. A successful convergence implies an alignment that meets the defined ICP error threshold, and no additional processing is required after that step.

The data in Table 5.2 illustrates the actual motions experienced by the camera during the acquisition of this data sequence, as seen from the coordinate frame of the world. These quantities are determined only after registration by completing the following cycle of homogeneous transformations for the general case of $k$ views:

$$ {}^{c_{m+1}}_{c_m}T = {}^{c_{m+1}}_{s_{m+1}}T \, {}^{s_{m+1}}_{m+1}T \, {}^{m+1}_{m}T \, {}^{m}_{s_m}T \, {}^{s_m}_{c_m}T, \qquad m \in \{1 \cdots k-1\} \tag{5.2} $$

| | | Angular Motions | | | | Translational Motions | | |
|---|---|---|---|---|---|---|---|---|
| Views | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
| $1 \rightarrow 2$ | 0.7989 | 0.6347 | 0.3022 | -0.3778 | 0.0171 | 0.0165 | 0.0044 | 0.0017 |
| $2 \rightarrow 3$ | 10.1536 | -1.0772 | 9.9341 | 1.7147 | 0.1223 | 0.1207 | 0.0153 | 0.0123 |
| $3 \rightarrow 4$ | 4.8283 | 2.6911 | 3.9510 | 0.7800 | 0.0898 | 0.0868 | 0.0206 | -0.0100 |

Continued on next page...

Table 5.2 – Continued from previous page...

| Views | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
|---|---|---|---|---|---|---|---|---|
| 4 → 5 | 13.0237 | -1.0261 | 12.6473 | 2.8293 | 0.1250 | 0.1179 | 0.0084 | 0.0405 |
| 5 → 6 | 4.9106 | 0.9491 | 4.8152 | 0.2095 | 0.0692 | 0.0685 | 0.0055 | -0.0079 |
| 6 → 7 | 10.6553 | -2.3560 | 9.9545 | 2.7902 | 0.0955 | 0.0908 | -0.0055 | 0.0289 |
| 7 → 8 | 3.4881 | 1.7309 | 2.9540 | 0.7135 | 0.0561 | 0.0561 | 0.0011 | 0.0009 |
| 8 → 9 | 3.3304 | -0.9922 | 3.0756 | 0.7788 | 0.0607 | 0.0602 | -0.0036 | 0.0069 |
| 9 → 10 | 9.2721 | 0.5606 | 8.9532 | 2.3915 | 0.0810 | 0.0800 | -0.0001 | 0.0131 |
| 10 → 11 | 1.2259 | 0.7167 | -0.0033 | -0.9945 | 0.0380 | 0.0371 | 0.0064 | -0.0050 |
| 11 → 12 | 11.5625 | -0.8155 | 10.9101 | 3.6700 | 0.1171 | 0.1151 | -0.0062 | 0.0208 |
| 12 → 13 | 1.8638 | 1.1775 | 1.4437 | -0.0445 | 0.0610 | 0.0606 | -0.0019 | -0.0062 |
| 13 → 14 | 12.6308 | 0.2541 | 12.3374 | 2.7275 | 0.1414 | 0.1405 | 0.0082 | 0.0141 |
| 14 → 15 | 3.7125 | 1.0928 | 3.5425 | 0.2363 | 0.0640 | 0.0640 | -0.0005 | 0.0029 |
| 15 → 16 | 10.9468 | -0.5938 | 10.6729 | 2.3089 | 0.1049 | 0.1033 | 0.0136 | 0.0121 |
| 16 → 17 | 5.0255 | 1.0153 | 4.1781 | 2.6397 | 0.0921 | 0.0911 | -0.0011 | -0.0135 |
| 17 → 18 | 9.9755 | -2.2223 | 9.7175 | -0.6240 | 0.1118 | 0.1083 | 0.0068 | 0.0271 |
| 18 → 19 | 8.2545 | 0.9390 | 8.0530 | 1.6196 | 0.1031 | 0.1022 | -0.0014 | 0.0133 |
| 19 → 20 | 15.0935 | -1.0254 | 14.8027 | 2.6433 | 0.1551 | 0.1531 | 0.0014 | 0.0246 |
| 20 → 21 | 5.2065 | 1.7076 | 4.3947 | 2.2762 | 0.0930 | 0.0925 | 0.0086 | 0.0055 |
| 21 → 22 | 10.3943 | -2.7735 | 9.8213 | 1.7563 | 0.1186 | 0.1138 | -0.0107 | 0.0316 |
| 22 → 23 | 11.9542 | 1.1180 | 11.8033 | 1.6522 | 0.0960 | 0.0950 | 0.0027 | 0.0136 |
| 23 → 24 | 12.3279 | 0.0593 | 12.0055 | 2.8118 | 0.1244 | 0.1235 | 0.0116 | 0.0086 |
| 24 → 25 | 8.0235 | -1.2134 | 6.8417 | 3.9427 | 0.1010 | 0.0970 | -0.0141 | 0.0244 |
| 25 → 26 | 10.2937 | -0.4803 | 10.1847 | 1.3748 | 0.1344 | 0.1328 | 0.0163 | 0.0126 |
| 26 → 27 | 8.7313 | 1.0446 | 8.3219 | 2.5069 | 0.0856 | 0.0845 | 0.0069 | 0.0115 |
| 27 → 28 | 9.3968 | 0.4581 | 9.3856 | 0.0765 | 0.1065 | 0.1054 | 0.0138 | -0.0049 |
| 28 → 29 | 11.5994 | -0.4474 | 10.7437 | 4.3142 | 0.1400 | 0.1375 | 0.0122 | 0.0231 |
| 29 → 30 | 5.5349 | -1.1353 | 5.3841 | 0.5482 | 0.0968 | 0.0967 | -0.0060 | 0.0000 |
| 30 → 31 | 7.4737 | -0.5507 | 6.5727 | 3.4852 | 0.1057 | 0.1045 | -0.0156 | -0.0016 |

Continued on next page...

Table 5.2 – Continued from previous page...

| Views | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
|---|---|---|---|---|---|---|---|---|
| 31 → 32 | 8.6288 | -0.8012 | 8.5741 | 0.4909 | 0.1241 | 0.1238 | 0.0027 | 0.0067 |
| 32 → 33 | 5.8564 | 0.8645 | 5.2401 | 2.5089 | 0.1090 | 0.1087 | -0.0039 | -0.0070 |
| 33 → 34 | 4.7975 | -0.4695 | 4.7608 | -0.3814 | 0.0736 | 0.0726 | 0.0014 | -0.0118 |
| 34 → 35 | 16.4956 | 0.4601 | 15.2464 | 6.3608 | 0.1442 | 0.1425 | -0.0027 | 0.0221 |
| 35 → 36 | 3.9290 | 0.0028 | 3.6844 | -1.3648 | 0.0568 | 0.0547 | 0.0030 | -0.0149 |
| 36 → 37 | 18.5559 | -0.3163 | 17.8388 | 5.0701 | 0.1366 | 0.1328 | 0.0005 | 0.0320 |
| 37 → 38 | 7.4475 | 0.5730 | 7.3862 | 0.8012 | 0.0510 | 0.0488 | 0.0100 | -0.0105 |
| 38 → 39 | 15.4254 | 0.1600 | 14.5853 | 5.0527 | 0.1268 | 0.1218 | -0.0046 | 0.0349 |
| 39 → 40 | 2.2272 | -1.3880 | 1.7179 | 0.2673 | 0.0743 | 0.0733 | -0.0076 | 0.0088 |
| 40 → 41 | 13.9207 | 0.1222 | 13.5285 | 3.3009 | 0.1346 | 0.1301 | 0.0053 | 0.0340 |
| 41 → 42 | 3.4569 | 0.5259 | 3.3923 | 0.4238 | 0.0656 | 0.0655 | 0.0041 | 0.0002 |
| 42 → 43 | 15.9694 | 0.5752 | 15.4242 | 4.1883 | 0.1467 | 0.1441 | 0.0076 | 0.0264 |
| 43 → 44 | 1.5924 | -0.5206 | 1.3495 | 0.6599 | 0.0820 | 0.0811 | -0.0022 | 0.0120 |
| 44 → 45 | 13.6728 | -0.2732 | 13.2999 | 3.1348 | 0.1351 | 0.1351 | 0.0003 | 0.0030 |
| 45 → 46 | 6.0697 | 1.1008 | 5.3719 | 2.6557 | 0.0713 | 0.0713 | 0.0017 | 0.0013 |
| 46 → 47 | 10.4648 | -1.6615 | 10.3105 | 0.5402 | 0.1169 | 0.1154 | 0.0116 | 0.0147 |
| 47 → 48 | 13.4204 | -0.0514 | 12.3072 | 5.3561 | 0.1074 | 0.0852 | 0.0636 | 0.0150 |
| 48 → 49 | 8.0996 | -0.6834 | 7.9939 | 1.0656 | 0.0788 | 0.0419 | 0.0636 | -0.0202 |
| 49 → 50 | 9.9481 | -0.4089 | 8.8195 | 4.5573 | 0.1231 | 0.1065 | -0.0572 | 0.0232 |
| median | 8.6054 | 0.0028 | 8.5741 | 1.7563 | 0.1031 | 0.0970 | 0.0027 | 0.0115 |
| $\mu$ | 8.7313 | -0.0561 | 8.1333 | 1.9479 | 0.0993 | 0.0961 | 0.0040 | 0.0097 |
| $\sigma$ | 4.4186 | 1.0970 | 4.3835 | 1.7852 | 0.0311 | 0.0313 | 0.0167 | 0.0146 |

Table 5.2: The actual motions experienced by the camera between adjacent views during data acquisition, as seen from the world coordinate frame. The units of measurement of angular motions are degrees, and those for translational motions are meters.

## 5.2 Registration Error Analysis

In this section, we present the intermediate qualitative results of our system immediately prior to global registration and surface reconstruction. As mentioned previously, our system maintains a record of all of the actions undertaken during the registration process. The correctness of a match is measured internally using the point–to–point distance that remains between each pair of point clouds reviewed during the procedure. However, further analysis of our records allows us to determine how inaccurate the initial position of each point set was before applying ICP, and the amount of rotation and translation required to achieve proper alignment.

Tables 5.3 and 5.5 summarize the amount of frame–to–frame error we observed after each stage of our procedure, and Tables 5.4 and 5.6 indicate the amount of absolute error that remained after each stage.

Similar to Table 3.3, the primary error measure we use for rotation throughout this section is arc length; it quantifies the magnitude of rotation around a single axis that will align two coordinate frames. It also allows the reader to easily determine which frames required less rotational correction by examining a single metric, rather than trying to discern an absolute order from a triplet of Euler angles. However, the Euler angles may be used in a complementary capacity to identify trends within each individual component of rotation.

| | Angular Motions | | | | Translational Motions | | | |
|---|---|---|---|---|---|---|---|---|
| **Views** | **Arc ($\epsilon$)** | **Roll ($\phi$)** | **Pitch ($\theta$)** | **Yaw ($\psi$)** | **$\rho$ (m)** | **x (m)** | **y (m)** | **z (m)** |
| $1 \leftarrow 2$ | 1.1686 | 0.3209 | 0.8013 | -0.7855 | 0.0172 | -0.0165 | -0.0008 | 0.0047 |
| $2 \leftarrow 3$ | 9.1681 | 2.4194 | 1.3145 | 8.7735 | 0.1224 | -0.1201 | 0.0137 | 0.0190 |
| $3 \leftarrow 4$ | 6.6925 | -2.6179 | -0.2944 | -6.1460 | 0.0898 | -0.0843 | 0.0217 | 0.0221 |
| $4 \leftarrow 5$ | 9.7647 | 2.1334 | -3.0114 | 8.9861 | 0.1250 | -0.1222 | 0.0154 | 0.0213 |
| $5 \leftarrow 6$ | 9.8052 | -2.7174 | 1.8815 | -9.2774 | 0.0691 | -0.0611 | 0.0322 | 0.0032 |
| $6 \leftarrow 7$ | 7.9353 | 2.5319 | -2.4852 | 7.0446 | 0.0955 | -0.0884 | 0.0360 | 0.0032 |
| $7 \leftarrow 8$ | 8.7072 | -1.4920 | 2.0664 | -8.3534 | 0.0561 | -0.0458 | 0.0324 | 0.0016 |
| $8 \leftarrow 9$ | 4.4900 | 0.5039 | -1.7175 | 4.1104 | 0.0608 | -0.0463 | 0.0393 | -0.0006 |
| $9 \leftarrow 10$ | 4.1997 | -0.4391 | 1.1482 | 4.0115 | 0.0811 | -0.0588 | 0.0557 | 0.0038 |
| $10 \leftarrow 11$ | 8.3497 | -0.2684 | 0.8548 | -8.3036 | 0.0380 | -0.0199 | 0.0319 | 0.0054 |

Continued on next page...

Table 5.3 – Continued from previous page...

| Views | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
|---|---|---|---|---|---|---|---|---|
| 11 ← 12 | 11.4578 | 0.2236 | -2.1876 | 11.2412 | 0.1171 | -0.0629 | 0.0987 | 0.0026 |
| 12 ← 13 | 9.8184 | -0.4955 | 2.3236 | -9.5373 | 0.0610 | -0.0271 | 0.0546 | -0.0023 |
| 13 ← 14 | 12.3113 | -0.5408 | -2.2755 | 12.0987 | 0.1414 | -0.0348 | 0.1363 | 0.0143 |
| 14 ← 15 | 10.3325 | 1.1138 | 2.8910 | -9.8302 | 0.0641 | -0.0227 | 0.0599 | -0.0002 |
| 15 ← 16 | 7.6236 | -1.8414 | -2.5633 | 6.9818 | 0.1049 | -0.0036 | 0.1031 | 0.0190 |
| 16 ← 17 | 5.8366 | -0.3571 | 1.4843 | -5.6382 | 0.0920 | 0.0136 | 0.0910 | -0.0009 |
| 17 ← 18 | 9.3546 | 1.9422 | -3.6257 | 8.3425 | 0.1118 | 0.0051 | 0.1112 | 0.0106 |
| 18 ← 19 | 4.1138 | -0.2703 | 2.1560 | -3.4985 | 0.1031 | 0.0139 | 0.1021 | 0.0037 |
| 19 ← 20 | 9.4279 | -1.1979 | -0.4333 | 9.3462 | 0.1551 | 0.0654 | 0.1404 | 0.0072 |
| 20 ← 21 | 11.7946 | 0.4904 | 1.7343 | -11.6491 | 0.0931 | 0.0419 | 0.0825 | 0.0102 |
| 21 ← 22 | 8.1419 | -2.7104 | -2.5446 | 7.3054 | 0.1186 | 0.0672 | 0.0977 | -0.0027 |
| 22 ← 23 | 2.9773 | 2.3686 | 0.9095 | -1.5392 | 0.0961 | 0.0711 | 0.0645 | 0.0033 |
| 23 ← 24 | 4.1137 | -0.2214 | 1.5698 | 3.7931 | 0.1243 | 0.1137 | 0.0489 | 0.0117 |
| 24 ← 25 | 10.2452 | -2.4975 | -0.0092 | -9.9367 | 0.1001 | 0.0848 | 0.0542 | -0.0073 |
| 25 ← 26 | 9.2319 | 0.6507 | -1.7045 | 9.0405 | 0.1344 | 0.1336 | 0.0008 | 0.0146 |
| 26 ← 27 | 3.271 | 0.1879 | 0.9779 | -3.1142 | 0.0856 | 0.0849 | 0.0052 | 0.0097 |
| 27 ← 28 | 2.5473 | 0.4055 | -1.7139 | 1.8344 | 0.1065 | 0.1002 | -0.0353 | 0.0072 |
| 28 ← 29 | 4.5238 | -1.1357 | 4.2320 | -1.1682 | 0.1400 | 0.1329 | -0.0400 | 0.0185 |
| 29 ← 30 | 2.6846 | -0.3245 | -1.6165 | -2.1142 | 0.0968 | 0.0827 | -0.0489 | -0.0118 |
| 30 ← 31 | 2.0971 | 0.6336 | 1.8997 | 0.6330 | 0.1057 | 0.0807 | -0.0665 | -0.0156 |
| 31 ← 32 | 2.9796 | -0.7293 | -0.8415 | 2.7692 | 0.1241 | 0.0865 | -0.0889 | -0.0029 |
| 32 ← 33 | 5.8373 | 1.9588 | -1.3927 | -5.3438 | 0.1089 | 0.0630 | -0.0886 | -0.0071 |
| 33 ← 34 | 3.3115 | -1.4335 | 0.9793 | -2.8324 | 0.0737 | 0.0246 | -0.0692 | -0.0057 |
| 34 ← 35 | 8.5661 | 1.9965 | 0.1858 | 8.3318 | 0.1442 | 0.0455 | -0.1368 | 0.0029 |
| 35 ← 36 | 9.9030 | -2.9830 | -0.6326 | -9.4065 | 0.0568 | -0.0009 | -0.0566 | -0.0044 |
| 36 ← 37 | 12.5756 | 2.5927 | 0.5363 | 12.3070 | 0.1365 | -0.0005 | -0.1363 | 0.0077 |
| 37 ← 38 | 7.7334 | -1.3706 | -1.0127 | -7.5315 | 0.0510 | -0.0195 | -0.0469 | 0.0045 |

Continued on next page...

Table 5.3 – Continued from previous page...

| Views | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
|---|---|---|---|---|---|---|---|---|
| 38 ← 39 | 6.1152 | 1.9943 | -0.4157 | 5.7590 | 0.1267 | -0.0332 | -0.1223 | 0.0018 |
| 39 ← 40 | 11.6661 | -1.5179 | 0.7765 | -11.5515 | 0.0743 | -0.0241 | -0.0697 | -0.0090 |
| 40 ← 41 | 10.0682 | 0.0463 | 0.6997 | 10.0441 | 0.1346 | -0.0845 | -0.1042 | 0.0104 |
| 41 ← 42 | 9.8664 | -0.7280 | -0.6807 | -9.8117 | 0.0656 | -0.0477 | -0.0450 | 0.0027 |
| 42 ← 43 | 13.5021 | 1.1618 | 0.1222 | 13.453 | 0.1467 | -0.1296 | -0.0677 | 0.0114 |
| 43 ← 44 | 15.7724 | -0.7354 | 0.4930 | -15.7508 | 0.0820 | -0.0640 | -0.0513 | -0.0005 |
| 44 ← 45 | 14.6076 | 1.0766 | -1.1324 | 14.5137 | 0.1351 | -0.1350 | -0.0047 | 0.0008 |
| 45 ← 46 | 9.7687 | -0.7300 | -0.4867 | -9.7262 | 0.0714 | -0.0702 | -0.0126 | 0.0021 |
| 46 ← 47 | 7.1966 | 2.1600 | 0.3520 | 6.8628 | 0.1169 | -0.1160 | 0.0098 | 0.0109 |
| 47 ← 48 | 2.2313 | 0.0412 | -1.7209 | -1.4203 | 0.1074 | -0.0802 | 0.0238 | 0.0673 |
| 48 ← 49 | 2.4125 | -1.0756 | 2.0985 | -0.5299 | 0.0787 | -0.0234 | 0.0495 | 0.0566 |
| 49 ← 50 | 1.9884 | -1.0739 | -1.5787 | -0.5404 | 0.1232 | -0.1099 | 0.0340 | -0.0440 |
| median | 7.5161 | -0.2684 | 0.1222 | -0.5404 | 0.1031 | -0.0199 | 0.01540 | 0.0033 |
| $\mu$ | 8.1419 | -0.0521 | -0.0324 | 0.2500 | 0.0993 | -0.0090 | 0.0072 | 0.0057 |
| $\sigma$ | 3.6812 | 1.5090 | 1.7112 | 8.0486 | 0.0314 | 0.0733 | 0.0710 | 0.01581 |

Table 5.3: Incremental rotations and translations required to align consecutive pairs of views. These measurements allow us to benchmark the local performance of the inertial sensor in providing initial conditions for our registration procedure; lower values indicate good initial conditions. They represent the error remaining after the initial estimation from the INS, but before applying our heuristic. The units of measurement of angular motions are degrees, and those for translational motions are meters.

On the other hand, error in position is indicated using the magnitude $\rho$ of the translation vector represented in Cartesian coordinates. The magnitude of the vector can be used in much the same way as arc length to quickly determine which frames required smaller corrections in position during registration. The components of the translation vector are also indicated so the reader may identify trends within each individual component of translation. All of the tables in this section indicate the amount of error remaining

after using the inertial navigation sensor for coarse registration, but before applying our heuristic.

Table 5.3 summarizes the amount of motion observed between adjacent frames in the artwork data sequence during intra-group registration. It is important to note that the motions enumerated here do not place the points at their final locations in the end model; they merely indicate the steps required between each pair of adjacent elements to meet that objective. The first image is selected as the reference for the initial alignment, so it remains unmodified, and we employ the same reference throughout this chapter. The choice of reference frame is made purely out of convenience, but any arbitrary frame would suffice.

Examining our data indicates that, the initial estimate (*i.e.,* measurements from the INS) of the orientation of a point set before registration is within roughly 8.1° of its correct global orientation, with a standard deviation of 3.7°. Generally, the amount of rotation observed to complete the alignment using ICP is quite small, indicating that the inertial sensor fares well at providing an accurate attitude and heading. Curiously, the technical sheet of the hardware indicates that this error should be smaller, and we speculate on the potential causes for this phenomenon later in this section. Similarly, the initial estimate of the position of a point set falls, on average, within 9.9 cm of its correct global location prior to applying our procedure, with a standard deviation of 3.1 cm. We feel that these results are acceptable overall, but that the error correction required for position is slightly higher than expected.

| | | **Angular Motions** | | | **Translational Motions** | | | |
|---|---|---|---|---|---|---|---|---|
| **Views** | **Arc ($\epsilon$)** | **Roll ($\phi$)** | **Pitch ($\theta$)** | **Yaw ($\psi$)** | **$\rho$ (m)** | **x (m)** | **y (m)** | **z (m)** |
| $1 \leftarrow 2$ | 1.1686 | 0.3209 | 0.8013 | -0.7855 | 0.0172 | -0.0165 | -0.0008 | 0.0047 |
| $1 \leftarrow 3$ | 8.6916 | 2.8591 | 2.0575 | 7.9987 | 0.1394 | -0.1365 | 0.0156 | 0.0233 |
| $1 \leftarrow 4$ | 2.7613 | 0.0041 | 2.0575 | 1.8417 | 0.2274 | -0.2216 | 0.0215 | 0.0461 |
| $1 \leftarrow 5$ | 11.155 | 2.4587 | -0.9798 | 10.8164 | 0.3514 | -0.3350 | 0.0733 | 0.0765 |
| $1 \leftarrow 6$ | 2.0974 | -0.1324 | 1.3089 | 1.6320 | 0.4145 | -0.4059 | 0.0474 | 0.0692 |
| $1 \leftarrow 7$ | 9.1465 | 2.5610 | -1.1700 | 8.6771 | 0.5081 | -0.4820 | 0.1365 | 0.0852 |
| $1 \leftarrow 8$ | 1.8099 | 1.2121 | 1.2784 | 0.4288 | 0.5581 | -0.5453 | 0.0957 | 0.0704 |
| $1 \leftarrow 9$ | 4.8847 | 1.8043 | -0.5288 | 4.5002 | 0.6125 | -0.5810 | 0.1745 | 0.0847 |
| $1 \leftarrow 10$ | 8.6554 | 1.3238 | 0.4940 | 8.5453 | 0.6878 | -0.6277 | 0.2699 | 0.0788 |

Continued on next page...

Table 5.4 – Continued from previous page...

| Views | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
|---|---|---|---|---|---|---|---|---|
| 1 ← 11 | 1.8326 | 0.9706 | 1.5346 | 0.2618 | 0.7161 | -0.6811 | 0.2079 | 0.0751 |
| 1 ← 12 | 11.5945 | 1.4745 | -0.8712 | 11.4566 | 0.8197 | -0.6870 | 0.4358 | 0.1002 |
| 1 ← 13 | 2.8312 | 1.1305 | 1.7079 | 1.9867 | 0.8664 | -0.7803 | 0.3699 | 0.0702 |
| 1 ← 14 | 14.0896 | 0.8958 | -0.8361 | 14.0299 | 0.9819 | -0.7169 | 0.6607 | 0.1166 |
| 1 ← 15 | 5.2180 | 2.1399 | 2.2196 | 4.2520 | 1.0338 | -0.8466 | 0.5898 | 0.0647 |
| 1 ← 16 | 11.1595 | 0.5512 | -0.6180 | 11.1259 | 1.1096 | -0.7686 | 0.7879 | 0.1401 |
| 1 ← 17 | 5.5840 | 0.2522 | 0.9234 | 5.5034 | 1.1599 | -0.8320 | 0.7989 | 0.1218 |
| 1 ← 18 | 14.3355 | 2.3261 | -2.7486 | 13.8225 | 1.2480 | -0.6931 | 1.0275 | 0.1459 |
| 1 ← 19 | 10.6674 | 2.2167 | -0.4476 | 10.4170 | 1.3223 | -0.7456 | 1.0848 | 0.1258 |
| 1 ← 20 | 19.8101 | 0.9170 | -1.2346 | 19.7410 | 1.4267 | -0.4931 | 1.3289 | 0.1620 |
| 1 ← 21 | 8.3097 | 1.6376 | 0.7099 | 8.1263 | 1.4770 | -0.7140 | 1.2855 | 0.1391 |
| 1 ← 22 | 15.5020 | -0.9948 | -2.0476 | 15.3527 | 1.5451 | -0.4708 | 1.4543 | 0.2249 |
| 1 ← 23 | 13.9343 | 1.4297 | -1.1641 | 13.7979 | 1.5847 | -0.4422 | 1.5133 | 0.1603 |
| 1 ← 24 | 17.6631 | 1.1279 | 0.3133 | 17.6277 | 1.6132 | -0.2316 | 1.5876 | 0.1685 |
| 1 ← 25 | 7.8475 | -1.4406 | 0.4939 | 7.6923 | 1.6397 | -0.4172 | 1.5693 | 0.2275 |
| 1 ← 26 | 16.8080 | -0.6946 | -0.9899 | 16.7707 | 1.6349 | -0.025 | 1.6187 | 0.2285 |
| 1 ← 27 | 13.6527 | -0.4517 | -0.0483 | 13.6453 | 1.6376 | -0.0319 | 1.6209 | 0.2310 |
| 1 ← 28 | 15.5902 | -0.0478 | -1.7477 | 15.4932 | 1.6084 | 0.1270 | 1.5874 | 0.2260 |
| 1 ← 29 | 14.6056 | -0.1478 | 2.4837 | 14.3236 | 1.5844 | 0.2106 | 1.5448 | 0.2823 |
| 1 ← 30 | 12.3802 | -1.5622 | 0.8236 | 12.2428 | 1.5453 | 0.2441 | 1.5009 | 0.2748 |
| 1 ← 31 | 13.1660 | -0.9210 | 2.7398 | 12.8241 | 1.4940 | 0.3321 | 1.4348 | 0.2512 |
| 1 ← 32 | 15.8249 | -1.5163 | 1.9397 | 15.6078 | 1.4368 | 0.4912 | 1.3249 | 0.2603 |
| 1 ← 33 | 10.3146 | 0.2692 | 0.3979 | 10.3044 | 1.3701 | 0.4349 | 1.2841 | 0.1975 |
| 1 ← 34 | 7.7096 | -1.1841 | 1.3900 | 7.4762 | 1.3108 | 0.3921 | 1.2293 | 0.2306 |
| 1 ← 35 | 15.9176 | 1.0262 | 1.7326 | 15.8061 | 1.2133 | 0.6108 | 1.0301 | 0.1948 |
| 1 ← 36 | 6.9095 | -2.2541 | 1.2445 | 6.3879 | 1.1608 | 0.4354 | 1.0481 | 0.2435 |
| 1 ← 37 | 18.8030 | 0.6549 | 2.2323 | 18.6726 | 1.0545 | 0.6461 | 0.8052 | 0.2148 |

Continued on next page...

Table 5.4 – Continued from previous page...

| Views | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
|---|---|---|---|---|---|---|---|---|
| $1 \leftarrow 38$ | 11.2600 | -1.0143 | 1.2862 | 11.1292 | 1.0066 | 0.5192 | 0.8309 | 0.2309 |
| $1 \leftarrow 39$ | 16.9498 | 1.1143 | 0.9658 | 16.8954 | 0.8937 | 0.5684 | 0.6597 | 0.2012 |
| $1 \leftarrow 40$ | 5.7415 | -0.6192 | 1.9458 | 5.3559 | 0.8214 | 0.3980 | 0.6847 | 0.2180 |
| $1 \leftarrow 41$ | 15.6468 | -0.2241 | 2.7236 | 15.4024 | 0.6967 | 0.4241 | 0.5007 | 0.2342 |
| $1 \leftarrow 42$ | 6.1117 | -1.4129 | 1.9649 | 5.5884 | 0.6390 | 0.2876 | 0.5177 | 0.2401 |
| $1 \leftarrow 43$ | 19.1868 | 0.2448 | 2.3619 | 19.0457 | 0.5214 | 0.2700 | 0.3738 | 0.2434 |
| $1 \leftarrow 44$ | 4.5005 | -1.1415 | 2.8327 | 3.2780 | 0.4629 | 0.0922 | 0.3786 | 0.2498 |
| $1 \leftarrow 45$ | 17.9187 | 0.6823 | 1.8959 | 17.8173 | 0.4226 | 0.0541 | 0.3433 | 0.2404 |
| $1 \leftarrow 46$ | 8.2337 | -0.3779 | 1.4972 | 8.0829 | 0.4199 | -0.0728 | 0.3318 | 0.2469 |
| $1 \leftarrow 47$ | 15.1593 | 1.9637 | 1.8836 | 14.9469 | 0.4581 | -0.1502 | 0.3570 | 0.2446 |
| $1 \leftarrow 48$ | 13.6078 | 1.9565 | 0.2117 | 13.4690 | 0.5442 | -0.2318 | 0.3772 | 0.3164 |
| $1 \leftarrow 49$ | 13.2282 | 0.8804 | 2.3271 | 13.0111 | 0.6213 | -0.2701 | 0.4187 | 0.3712 |
| $1 \leftarrow 50$ | 12.4728 | -0.2161 | 0.7567 | 12.4466 | 0.6730 | -0.3736 | 0.4429 | 0.3423 |
| median | 10.8663 | 0.5512 | 0.9658 | 11.1292 | 0.9942 | -0.2216 | 0.6607 | 0.2012 |
| $\mu$ | 11.2600 | 0.4501 | 0.7777 | 10.5076 | 0.9849 | -0.1528 | 0.7711 | 0.1795 |
| $\sigma$ | 5.1592 | 1.2563 | 1.3901 | 5.4691 | 0.4545 | 0.4550 | 0.5329 | 0.0856 |

Table 5.4: Absolute rotations and translations required to position each view in its correct position in the rough end model. These measurements allow us to gauge the global performance of the inertial sensor in localizing the camera over larger distances and complex motions; lower values indicate good initial conditions. They represent the error remaining after the initial estimation from the INS, but before applying our heuristic. The units of measurement of angular motions are degrees, and those for translational motions are meters.

In Table 5.4 we present equivalent results (*i.e.*, corrections needed after the initial estimate from the inertial sensor) for when the alignment of each view is determined with respect to a single reference frame. Once again, the reference we used to determine these measurements is the first element of the data sequence, for reasons we already enumer-

ated. Essentially, these values indicate the absolute motions applied to each individual point cloud (*i.e.*, camera view or viewpoint) in order to produce the rough global solution by compounding the motions from the previous table.

These values show that on average, each view has to rotate by approximately 11.3° before arriving at its final position, with a standard deviation of 5.2°. This result is still acceptable because the fact that no views were rejected shows that our heuristics are robust enough to correct most errors associated with orientation. Since we know that the attitude and heading of the inertial sensor is drift free, the larger values seen here are likely a result of the interaction of the small errors already present in each step.

Likewise, the average amount of translation required to achieve proper localization is 98 cm, with a standard deviation of 45 cm. The most observable trend when examining the magnitude of the translations is that it increases linearly, peaks about halfway through the sequence, and decreases again. This is consistent with the motion of the apparatus during acquisition of a closed–loop sequence where the maximum distance from the start is achieved at the halfway mark. Otherwise, we feel that the values of the translations are higher than expected, but this did not appear to cause our system any difficulty during registration.

Table 5.5 illustrates the amount of motion required to align adjacent groups. These measurements illustrate the amount of motion that is required in each individual step to position a group in its final location in the rough end model. The reader should note that these results are somewhat dependent on the aggressiveness with which the grouping scheme is executed. However, our system utilizes the most conservative application of our grouping scheme, and as a result the values observed here closely reflect those in Table 5.3 because each view acts as the reference frame of its own group.

| | | **Angular Motions** | | | **Translational Motions** | | | |
|---|---|---|---|---|---|---|---|---|
| **View** | **Arc ($\epsilon$)** | **Roll ($\phi$)** | **Pitch ($\theta$)** | **Yaw ($\psi$)** | **$\rho$ (m)** | **x (m)** | **y (m)** | **z (m)** |
| $1 \leftarrow 2$ | 0 | 0 | 0 | 0 | 0.0231 | -0.0230 | -0.0025 | 0 |
| $2 \leftarrow 3$ | 9.266 | 2.4681 | 1.3768 | 8.8552 | 0.1234 | -0.1211 | 0.0139 | 0.0194 |
| $3 \leftarrow 4$ | 6.6495 | -2.5883 | -0.3210 | -6.1099 | 0.0903 | -0.0847 | 0.0218 | 0.0226 |
| $4 \leftarrow 5$ | 9.8325 | 2.1809 | -3.0357 | 9.0383 | 0.1256 | -0.1226 | 0.0157 | 0.0221 |
| $5 \leftarrow 6$ | 9.8285 | -2.6977 | 1.9037 | -9.3035 | 0.0688 | -0.0608 | 0.0321 | 0.0036 |
| $6 \leftarrow 7$ | 7.9006 | 2.5002 | -2.4854 | 7.0176 | 0.0952 | -0.0882 | 0.0356 | 0.0027 |

Continued on next page...

Table 5.5 – Continued from previous page...

| View | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
|---|---|---|---|---|---|---|---|---|
| 7 ← 8 | 8.8555 | -1.5026 | 2.0652 | -8.507 | 0.0543 | -0.0443 | 0.0313 | 0.0013 |
| 8 ← 9 | 4.5375 | 0.5724 | -1.7068 | 4.1568 | 0.0612 | -0.0467 | 0.0396 | -0.0005 |
| 9 ← 10 | 4.1367 | -0.4240 | 1.1762 | 3.9390 | 0.0801 | -0.0582 | 0.0549 | 0.0038 |
| 10 ← 11 | 8.3158 | -0.2513 | 0.8121 | -8.2741 | 0.0385 | -0.0202 | 0.0323 | 0.0057 |
| 11 ← 12 | 11.468 | 0.2271 | -2.1911 | 11.2507 | 0.1172 | -0.0629 | 0.0989 | 0.0026 |
| 12 ← 13 | 9.7616 | -0.4155 | 2.3044 | -9.4856 | 0.0620 | -0.0275 | 0.0555 | -0.0021 |
| 13 ← 14 | 12.4028 | -0.4819 | -2.2998 | 12.1887 | 0.1426 | -0.0350 | 0.1375 | 0.0142 |
| 14 ← 15 | 10.2433 | 1.2041 | 2.8752 | -9.7285 | 0.0657 | -0.0231 | 0.0615 | 0.0001 |
| 15 ← 16 | 7.6797 | -1.8024 | -2.5798 | 7.0469 | 0.1055 | -0.0034 | 0.1037 | 0.0193 |
| 16 ← 17 | 5.8950 | -0.4615 | 1.5455 | -5.6765 | 0.0904 | 0.0132 | 0.0894 | -0.0022 |
| 17 ← 18 | 9.4179 | 1.8790 | -3.6912 | 8.3997 | 0.1122 | 0.0054 | 0.1115 | 0.0111 |
| 18 ← 19 | 4.0449 | -0.1239 | 2.1433 | -3.4307 | 0.1034 | 0.0140 | 0.1024 | 0.0040 |
| 19 ← 20 | 9.4401 | -1.1905 | -0.4352 | 9.3593 | 0.1553 | 0.0655 | 0.1406 | 0.0072 |
| 20 ← 21 | 11.9337 | 0.4650 | 1.7412 | -11.7902 | 0.0916 | 0.0411 | 0.0812 | 0.0100 |
| 21 ← 22 | 7.9312 | -2.8112 | -2.4911 | 7.0481 | 0.1153 | 0.0648 | 0.0954 | -0.0022 |
| 22 ← 23 | 2.9773 | 2.3686 | 0.9095 | -1.5392 | 0.0961 | 0.0711 | 0.0645 | 0.0033 |
| 23 ← 24 | 4.1279 | -0.2152 | 1.5621 | 3.8120 | 0.1246 | 0.1140 | 0.0490 | 0.0117 |
| 24 ← 25 | 10.0217 | -2.4190 | 0.0002 | -9.7261 | 0.1036 | 0.0870 | 0.0557 | -0.0083 |
| 25 ← 26 | 9.2669 | 0.6483 | -1.7777 | 9.0621 | 0.1349 | 0.1341 | 0.0008 | 0.0145 |
| 26 ← 27 | 3.3213 | 0.1739 | 0.9804 | -3.1671 | 0.0853 | 0.0846 | 0.0052 | 0.0096 |
| 27 ← 28 | 0 | 0 | 0 | 0 | 0.0843 | 0.0754 | -0.0353 | 0.0131 |
| 28 ← 29 | 4.5086 | -1.1182 | 4.2254 | -1.1484 | 0.1402 | 0.1331 | -0.0400 | 0.0183 |
| 29 ← 30 | 3.6396 | -1.2269 | -2.2627 | -2.5494 | 0.0942 | 0.0801 | -0.0492 | -0.0057 |
| 30 ← 31 | 1.9448 | 0.5369 | 1.7246 | 0.7292 | 0.1070 | 0.0819 | -0.0670 | -0.0160 |
| 31 ← 32 | 3.0338 | -0.6553 | -0.7860 | 2.8605 | 0.1245 | 0.0869 | -0.0891 | -0.0032 |
| 32 ← 33 | 5.8489 | 1.9488 | -1.4192 | -5.3536 | 0.1090 | 0.0631 | -0.0886 | -0.0072 |
| 33 ← 34 | 0 | 0 | 0 | 0 | 0.0981 | 0.0361 | -0.0893 | -0.0186 |

Continued on next page...

Table 5.5 – Continued from previous page...

| View | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
|---|---|---|---|---|---|---|---|---|
| 34 ← 35 | 8.5735 | 1.9973 | 0.1892 | 8.3392 | 0.1443 | 0.0455 | -0.1369 | 0.0030 |
| 35 ← 36 | 9.9030 | -2.9830 | -0.6326 | -9.4065 | 0.0568 | -0.0009 | -0.0566 | -0.0044 |
| 36 ← 37 | 12.5756 | 2.5927 | 0.5363 | 12.3070 | 0.1365 | -0.0005 | -0.1363 | 0.0077 |
| 37 ← 38 | 7.7334 | -1.3706 | -1.0127 | -7.5315 | 0.0510 | -0.0195 | -0.0469 | 0.0045 |
| 38 ← 39 | 6.1173 | 1.9945 | -0.4148 | 5.7612 | 0.1268 | -0.0332 | -0.1224 | 0.0018 |
| 39 ← 40 | 11.6252 | -1.4898 | 0.7998 | -11.5124 | 0.0746 | -0.0243 | -0.0700 | -0.0086 |
| 40 ← 41 | 10.1332 | 0.1577 | 0.6061 | 10.1147 | 0.1351 | -0.0848 | -0.1046 | 0.0106 |
| 41 ← 42 | 9.9200 | -0.7033 | -0.7108 | -9.8653 | 0.0651 | -0.0473 | -0.0446 | 0.0026 |
| 42 ← 43 | 13.5263 | 1.1908 | 0.0711 | 13.4745 | 0.1468 | -0.1297 | -0.0677 | 0.0115 |
| 43 ← 44 | 15.7700 | -0.7067 | 0.4850 | -15.7498 | 0.0820 | -0.0641 | -0.0512 | -0.0002 |
| 44 ← 45 | 14.6421 | 1.0817 | -1.1400 | 14.5472 | 0.1354 | -0.1353 | -0.0046 | 0.0008 |
| 45 ← 46 | 9.7780 | -0.7609 | -0.3809 | -9.7384 | 0.0714 | -0.0702 | -0.0127 | 0.0021 |
| 46 ← 47 | 7.1966 | 2.1600 | 0.3520 | 6.8628 | 0.1169 | -0.1160 | 0.0098 | 0.0109 |
| 47 ← 48 | 0 | 0 | 0 | 0 | 0.1223 | -0.1006 | 0.0269 | 0.0642 |
| 48 ← 49 | 0 | 0 | 0 | 0 | 0.0907 | -0.0152 | 0.0561 | 0.0696 |
| median | 8.1235 | 0 | 0 | 0 | 0.1034 | -0.0199 | 0.0148 | 0.0035 |
| $\mu$ | 7.4110 | -0.0011 | -0.0289 | 0.3455 | 0.1005 | -0.0076 | 0.0064 | 0.0069 |
| $\sigma$ | 4.0161 | 1.5008 | 1.6568 | 8.1202 | 0.0309 | 0.07256 | 0.07226 | 0.01535 |

Table 5.5: Incremental rotations and translations required to align consecutive pairs of groups. These values may depend on the aggressiveness of the grouping scheme employed, and reflect the most conservative grouping scheme that is employed by our system. They represent the error remaining after the initial estimation from the INS, but before applying our heuristic. The units of measurement of angular motions are degrees, and those for translational motions are meters.

A review of Table 5.5 confirms that on average only 7.4° of motion is required for inter-group alignment, with a standard deviation of 4.0°. The average correction required for the position of a group was determined to be 10 cm, with a standard deviation of 3.1 cm.

Again, the amount of rotation observed for alignment is quite small, indicating that the inertial sensor fares well at providing an accurate attitude and heading at each camera location, but we still feel there could be some improvement in the position estimates. Regardless, our heuristics were successful in merging this data to the correct global solution.

There are a few cases in these results where the rotational correction is listed as zero. The explanation for this phenomenon is that those consecutive frames were simply recorded during a pause in the rotation, possibly even a pause in the translation as well. Low values in the corresponding translational correction would be indicative of the latter case, and are likely a result of drift. Also all of the values in this table and all subsequent ones are rounded to four decimal places.

Lastly, 5.6 enumerates the motions of each group determined with respect to a single reference frame. The reference frame here is the first group, which is consistent with our previous selections for reasons mentioned earlier. The values in this table indicate how far each group moves before it arrives at its final position in the rough model produced by inter–group registration, and they retain their dependence on the grouping scheme, like the previous case.

These values indicate that globally, a group is rotated by 11.8° on average, with a standard deviation of 5.6°. Likewise, the average translation applied to a group is 96 cm, with a standard deviation of 48 cm. These results appear to be consistent with those of Table 5.4 since each view acts as the reference of its own group according to our grouping scheme. Reviewing the magnitude of translation in this table also reflects the same trend where the maximum distance that a group needs to traverse is observed halfway through the acquisition sequence. While it is clear that our system is capable of handling corrections as large as these, we feel that there is still room for improvement.

| | | **Angular Motions** | | | **Translational Motions** | | | |
|---|---|---|---|---|---|---|---|---|
| **View** | **Arc ($\epsilon$)** | **Roll ($\phi$)** | **Pitch ($\theta$)** | **Yaw ($\psi$)** | **$\rho$ (m)** | **x (m)** | **y (m)** | **z (m)** |
| $1 \leftarrow 2$ | 0 | 0 | 0 | 0 | 0.0231 | -0.0230 | -0.0025 | 0 |
| $1 \leftarrow 3$ | 9.2660 | 2.4681 | 1.3768 | 8.8552 | 0.1461 | -0.1441 | 0.0150 | 0.0188 |
| $1 \leftarrow 4$ | 3.0491 | -0.2810 | 1.3108 | 2.7355 | 0.2342 | -0.2295 | 0.0205 | 0.0422 |
| $1 \leftarrow 5$ | 12.1164 | 2.1094 | -1.6971 | 11.7800 | 0.3590 | -0.3435 | 0.0739 | 0.0739 |
| $1 \leftarrow 6$ | 2.6447 | -0.3424 | 0.5683 | 2.5585 | 0.4219 | -0.4140 | 0.0466 | 0.0665 |

Continued on next page...

Table 5.6 – Continued from previous page...

| View | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
|---|---|---|---|---|---|---|---|---|
| $1 \leftarrow 7$ | 10.0557 | 2.2297 | -1.8795 | 9.5881 | 0.5153 | -0.4901 | 0.1360 | 0.0824 |
| $1 \leftarrow 8$ | 1.6164 | 0.9798 | 0.5342 | 1.1740 | 0.5634 | -0.5518 | 0.0916 | 0.0671 |
| $1 \leftarrow 9$ | 5.6871 | 1.5885 | -1.2447 | 5.3000 | 0.6181 | -0.5881 | 0.1718 | 0.0814 |
| $1 \leftarrow 10$ | 9.3346 | 1.0748 | -0.1751 | 9.2694 | 0.6923 | -0.6347 | 0.2662 | 0.0749 |
| $1 \leftarrow 11$ | 1.5310 | 0.8377 | 0.7933 | 1.0122 | 0.7208 | -0.6875 | 0.2041 | 0.0719 |
| $1 \leftarrow 12$ | 12.4006 | 1.2038 | -1.5762 | 12.2251 | 0.8242 | -0.6941 | 0.4336 | 0.0973 |
| $1 \leftarrow 13$ | 3.1204 | 1.0313 | 0.9474 | 2.7971 | 0.8714 | -0.7869 | 0.3682 | 0.0670 |
| $1 \leftarrow 14$ | 15.0478 | 0.7266 | -1.5909 | 14.9363 | 0.9875 | -0.7232 | 0.6628 | 0.1133 |
| $1 \leftarrow 15$ | 5.8444 | 2.1891 | 1.4295 | 5.2549 | 1.0408 | -0.8525 | 0.5940 | 0.0607 |
| $1 \leftarrow 16$ | 12.2941 | 0.5457 | -1.4275 | 12.1924 | 1.1169 | -0.7731 | 0.7944 | 0.1366 |
| $1 \leftarrow 17$ | 6.5388 | 0.2226 | 0.1788 | 6.5330 | 1.1662 | -0.8380 | 0.8025 | 0.1172 |
| $1 \leftarrow 18$ | 15.5394 | 2.1258 | -3.5468 | 14.9168 | 1.2543 | -0.6973 | 1.0327 | 0.1437 |
| $1 \leftarrow 19$ | 11.8696 | 2.2069 | -1.2718 | 11.5696 | 1.3289 | -0.7487 | 1.0912 | 0.1214 |
| $1 \leftarrow 20$ | 21.0379 | 0.7808 | -2.0486 | 20.9107 | 1.4336 | -0.4949 | 1.3362 | 0.1577 |
| $1 \leftarrow 21$ | 9.3002 | 1.6474 | -0.1050 | 9.1513 | 1.4831 | -0.7208 | 1.2891 | 0.1348 |
| $1 \leftarrow 22$ | 16.3793 | -1.1872 | -2.7965 | 16.1260 | 1.5490 | -0.4860 | 1.4537 | 0.2236 |
| $1 \leftarrow 23$ | 14.7669 | 1.2577 | -1.9180 | 14.5676 | 1.5879 | -0.4573 | 1.5123 | 0.1588 |
| $1 \leftarrow 24$ | 18.4427 | 0.9116 | -0.4356 | 18.4118 | 1.6153 | -0.2460 | 1.5878 | 0.1665 |
| $1 \leftarrow 25$ | 8.8071 | -1.4469 | -0.2751 | 8.6868 | 1.6420 | -0.4237 | 1.5707 | 0.2223 |
| $1 \leftarrow 26$ | 17.8897 | -0.8246 | -1.8210 | 17.7917 | 1.6367 | -0.0299 | 1.6211 | 0.2237 |
| $1 \leftarrow 27$ | 14.6434 | -0.5485 | -0.8834 | 14.6109 | 1.6390 | -0.0386 | 1.6228 | 0.2263 |
| $1 \leftarrow 28$ | 14.6434 | -0.5485 | -0.8834 | 14.6109 | 1.6059 | 0.0368 | 1.5875 | 0.2394 |
| $1 \leftarrow 29$ | 13.9722 | -1.6497 | 3.3309 | 13.4232 | 1.5743 | 0.1204 | 1.5430 | 0.2883 |
| $1 \leftarrow 30$ | 11.4180 | -3.0206 | 0.9927 | 10.9417 | 1.5304 | 0.1431 | 1.4911 | 0.3135 |
| $1 \leftarrow 31$ | 12.2111 | -2.4738 | 2.7533 | 11.5794 | 1.4727 | 0.2344 | 1.4251 | 0.2882 |
| $1 \leftarrow 32$ | 14.975 | -2.9872 | 2.0880 | 14.4729 | 1.4090 | 0.3961 | 1.3192 | 0.2969 |
| $1 \leftarrow 33$ | 9.2945 | -1.2185 | 0.3832 | 9.2024 | 1.3370 | 0.3416 | 1.2708 | 0.2368 |

Continued on next page...

Table 5.6 – Continued from previous page...

| View | Arc ($\epsilon$) | Roll ($\phi$) | Pitch ($\theta$) | Yaw ($\psi$) | $\rho$ (m) | x (m) | y (m) | z (m) |
|---|---|---|---|---|---|---|---|---|
| $1 \leftarrow 34$ | 9.2945 | -1.2185 | 0.3832 | 9.2024 | 1.2594 | 0.3777 | 1.1815 | 0.2182 |
| $1 \leftarrow 35$ | 17.5668 | 0.8472 | 0.7451 | 17.5363 | 1.1622 | 0.5899 | 0.9843 | 0.1840 |
| $1 \leftarrow 36$ | 8.4404 | -2.2691 | 0.2410 | 8.1218 | 1.1097 | 0.4221 | 1.0001 | 0.2303 |
| $1 \leftarrow 37$ | 20.4377 | 0.4267 | 1.2549 | 20.3998 | 1.0039 | 0.6229 | 0.7606 | 0.2034 |
| $1 \leftarrow 38$ | 12.9157 | -1.1122 | 0.2873 | 12.8620 | 0.9561 | 0.5019 | 0.7839 | 0.2186 |
| $1 \leftarrow 39$ | 18.6526 | 0.9168 | -0.0172 | 18.6301 | 0.8440 | 0.5465 | 0.6143 | 0.1907 |
| $1 \leftarrow 40$ | 7.2256 | -0.5879 | 0.9658 | 7.1317 | 0.7717 | 0.3858 | 0.6357 | 0.2062 |
| $1 \leftarrow 41$ | 17.3279 | -0.2516 | 1.6601 | 17.2433 | 0.6468 | 0.4045 | 0.4540 | 0.2205 |
| $1 \leftarrow 42$ | 7.5444 | -1.2355 | 0.8816 | 7.3809 | 0.5893 | 0.2761 | 0.4693 | 0.2255 |
| $1 \leftarrow 43$ | 20.8876 | 0.1947 | 1.2163 | 20.8537 | 0.4715 | 0.2480 | 0.3290 | 0.2292 |
| $1 \leftarrow 44$ | 5.4560 | -0.8496 | 1.7085 | 5.0991 | 0.4135 | 0.08330 | 0.3299 | 0.2350 |
| $1 \leftarrow 45$ | 19.6813 | 0.6888 | 0.7272 | 19.6604 | 0.3760 | 0.03280 | 0.2981 | 0.2268 |
| $1 \leftarrow 46$ | 9.9312 | -0.2051 | 0.4523 | 9.9179 | 0.3771 | -0.0868 | 0.2836 | 0.2329 |
| $1 \leftarrow 47$ | 16.9048 | 2.0105 | 0.8256 | 16.7800 | 0.4231 | -0.1697 | 0.3103 | 0.2323 |
| $1 \leftarrow 48$ | 16.9048 | 2.0105 | 0.8256 | 16.7800 | 0.5241 | -0.2703 | 0.3372 | 0.2965 |
| $1 \leftarrow 49$ | 19.9048 | 2.0105 | 0.8256 | 16.7800 | 0.6085 | -0.2855 | 0.3933 | 0.3661 |
| median | 12.1638 | 0.4862 | 0.3832 | 11.6797 | 0.9718 | -0.1996 | 0.6493 | 0.1971 |
| $\mu$ | 11.7670 | 0.2288 | 0.0853 | 11.4909 | 0.9570 | -0.1702 | 0.7625 | 0.1735 |
| $\sigma$ | 5.6346 | 1.4430 | 1.4127 | 5.6210 | 0.4767 | 0.4451 | 0.5409 | 0.0851 |

Table 5.6: Absolute rotations and translations required to position each group in its correct position in the rough end model. These values may depend on the aggressiveness of the grouping scheme employed, and reflect the most conservative grouping scheme available in our system. They represent the error remaining after the initial estimation from the INS, but before applying our heuristic. The units of measurement of angular motions are degrees, and those for translational motions are meters.

Of note in Table 5.6 is the first relation, which indicates a rotational correction of zero between the first and second group. This result is simply caused by the fact that the op-

erator failed to move the camera rig until the first two frames had already been recorded, as evidenced by the low value of translational correction in the same comparison.

As was already mentioned, these internal results allow us to benchmark the performance of a couple of aspects of our 3D scanning system, with the foremost being the accuracy of the INS measurements supplied to our registration module. The measurements of the corrections applied between adjacent views and groups both indicate that the initial conditions are overall quite good in this case, which implies that the accuracy of the INS is quite good over short distances.

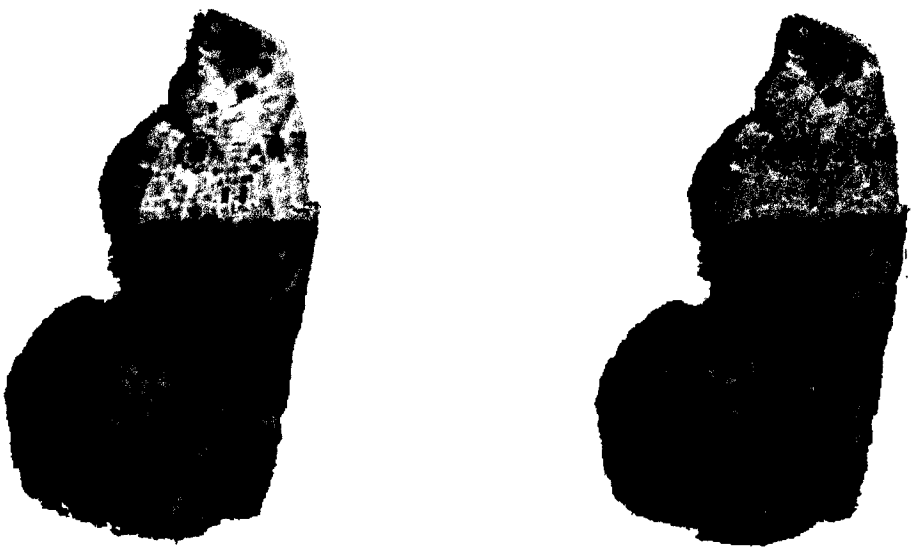
Conversely, the corrections required to localize a view or group in a single global reference frame are higher overall than the previous case. However, the orientation and position estimates exhibit different levels of performance in this case. According to specifications, the dynamic accuracy of the attitude of the Xsens MTi is $\approx 2°$ RMS, which is lower than most of the practical values we observed, but this value may be slightly understated by the manufacturer. We also speculate that the higher values for angular motion could reasonably be attributed to the interaction of errors already present in each step of the compound transformation that moves a view or group into the reference coordinate frame, which is expected. The increased values in translational motion appear much higher than expected, and could be the result of several different factors including but not limited to environmental noise and accelerometer drift, despite our best efforts to limit the impact of these effects.

Fortunately, the results presented in these tables also indicate that our registration module is robust enough to compensate for errors of the magnitude encountered in this data set because none of the views or groups were rejected. However, these intermediate results provide no information about the actual correctness or quality of the model produced by our scanning system relative to the subject material. Consequently, we desire a ground truth model with which we may compare the completed model generated by our 3D scanning system.

## 5.3 Ground Truth Evaluation

Once completed, the model generated by our 3D scanning system should be a fully textured polygonal surface representation of the subject, as shown in Figure 5.2(a). While the model generated by our system appears visually accurate, we still require some means of quantifying any error not visible to the naked eye. With this in mind, we use a commercial laser scanner to digitize a "ground truth" model of the subject because laser

range finding is currently the most resolute technique available. We utilize the Minolta Konica VIVID 910 triangulation laser digitizer in conjunction with a turntable for this purpose. The ground truth model of the artwork, shown in Figure 5.2(b), was acquired in a controlled indoor laboratory environment using a standard desktop computer.



(a) (b)

Figure 5.2: Models of the artwork data set: (a) the final model constructed using our 3D scanning system, and (b) the "ground truth" captured using a triangulation laser digitizer.

A physical comparison of these two models is performed using Geomagic, third–party commercial software which features a toolkit specifically for comparing three–dimensional models. The comparison tool automatically aligns two models, then determines the point–to–surface distances from one model to the second reference model. The purpose of the "ground truth" model in this task is to act as the reference sample with which we wish to compare our final product.

Typical results obtained using this tool are portrayed in Figure 5.3. The figure illustrates the ground truth model coded using a colour scale to indicate the deviation of our model from the reference sample in each location. Warm colours (*i.e.*, oranges and reds) indicate locations where our model extends above the surface of the ground truth model, and cold colours (*i.e.*, teals and blues) indicate locations where it extends below the surface of the ground truth model.

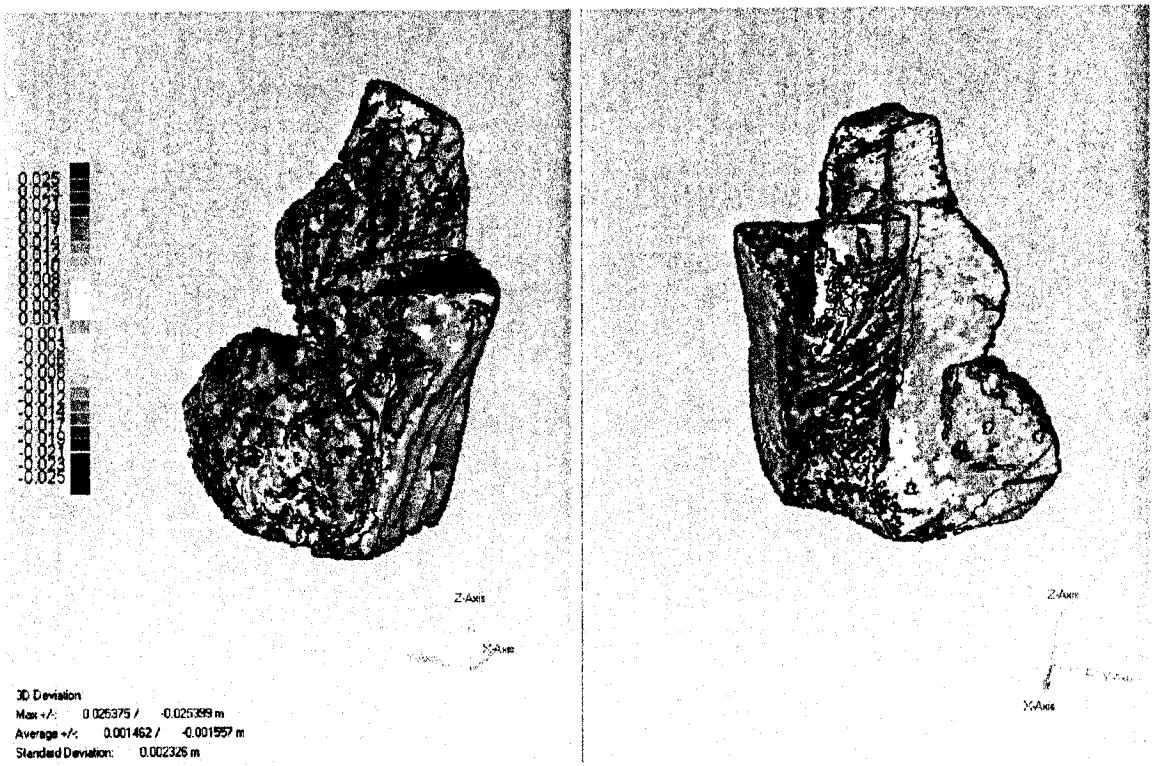



Figure 5.3: Results of the comparison of our full final model with the ground truth reference material.

The scale for this particular result is indicated on the left and ranges from -2.7 mm to 2.7 mm. Further examination shows that much of the surface of our model falls within $\pm 0.1$ mm of the ground truth, as indicated by the green and yellow regions. Overall, the maximum deviation for our completed model ranges from -2.5 mm to 2.5 mm, and the average ranges from -1.6 mm to 1.5 mm, which we feel is a very good result. The reason we can claim that this is a good result has to do with the local correctness of the shape of the model, and not the fine surface details as one might expect. In this case, low surface deviation indicates that surfaces that should coincide appropriately, and are aligned as such. If there were a problem with the shape of the model produced by our system, there would be a significant amount surface deviation.

Alternatively, we can perform the same operation to determine the error present in each individual view, which allows us to easily identify any views that are disproportionately erroneous, and therefore introduce the most deviation into the final product. Table 5.7 summarizes the surface deviations of all 50 views in the artwork data set. The

same technique can be applied to each individual group, and those results are compiled in Table 5.8.

| | Absolute Errors | | | | |
|---|---|---|---|---|---|
| **View** | **Max (m)** | **Min (m)** | **$+\mu$ (m)** | **$-\mu$ (m)** | **$\sigma$ (m)** |
| 1 | 0.011412 | -0.010434 | 0.001812 | -0.002610 | 0.001986 |
| 2 | 0.014168 | -0.011872 | 0.001757 | -0.002606 | 0.002037 |
| 3 | 0.015260 | -0.009471 | 0.001481 | -0.002620 | 0.002015 |
| 4 | 0.011183 | -0.012449 | 0.001481 | -0.002581 | 0.002099 |
| 5 | 0.012873 | -0.011856 | 0.001794 | -0.002631 | 0.002439 |
| 6 | 0.011492 | -0.013471 | 0.001652 | -0.002571 | 0.002372 |
| 7 | 0.014202 | -0.011487 | 0.001933 | -0.002592 | 0.002629 |
| 8 | 0.020201 | -0.011433 | 0.001977 | -0.002515 | 0.002668 |
| 9 | 0.017309 | -0.014461 | 0.001917 | -0.002476 | 0.002651 |
| 10 | 0.023957 | -0.012613 | 0.002162 | -0.002448 | 0.003000 |
| 11 | 0.024449 | -0.010863 | 0.002139 | -0.002462 | 0.002900 |
| 12 | 0.014117 | -0.014237 | 0.002040 | -0.002445 | 0.002836 |
| 13 | 0.023198 | -0.015785 | 0.002318 | -0.002166 | 0.002993 |
| 14 | 0.024442 | -0.010529 | 0.002179 | -0.001726 | 0.002603 |
| 15 | 0.019723 | -0.025181 | 0.002003 | -0.001731 | 0.002560 |
| 16 | 0.012391 | -0.014328 | 0.002006 | -0.001398 | 0.002251 |
| 17 | 0.019889 | -0.013143 | 0.002445 | -0.001572 | 0.002620 |
| 18 | 0.013535 | -0.024917 | 0.002561 | -0.001658 | 0.002766 |
| 19 | 0.014951 | -0.015190 | 0.002608 | -0.001516 | 0.002551 |
| 20 | 0.015409 | -0.013928 | 0.002874 | -0.001532 | 0.002694 |
| 21 | 0.019034 | -0.010215 | 0.002740 | -0.001582 | 0.002724 |
| 22 | 0.018654 | -0.011284 | 0.002930 | -0.001714 | 0.002779 |
| 23 | 0.014455 | -0.010086 | 0.003224 | -0.001811 | 0.003009 |
| 24 | 0.018120 | -0.025391 | 0.003317 | -0.002170 | 0.003346 |

Continued on next page...

Table 5.7 – Continued from previous page...

| View | Max (m) | Min (m) | $+\mu$ (m) | $-\mu$ (m) | $\sigma$ (m) |
|---|---|---|---|---|---|
| 25 | 0.014701 | -0.012602 | 0.002854 | -0.001863 | 0.002896 |
| 26 | 0.014975 | -0.011357 | 0.002442 | -0.001863 | 0.002796 |
| 27 | 0.015513 | -0.014372 | 0.002251 | -0.001943 | 0.002775 |
| 28 | 0.013730 | -0.013302 | 0.001798 | -0.001955 | 0.002484 |
| 29 | 0.012877 | -0.025111 | 0.001672 | -0.002067 | 0.002506 |
| 30 | 0.014273 | -0.017350 | 0.001629 | -0.002064 | 0.002430 |
| 31 | 0.015370 | -0.021011 | 0.001573 | -0.002172 | 0.002402 |
| 32 | 0.013405 | -0.016314 | 0.001471 | -0.002339 | 0.002409 |
| 33 | 0.022495 | -0.020302 | 0.001412 | -0.002355 | 0.002450 |
| 34 | 0.015269 | -0.017125 | 0.001440 | -0.002329 | 0.002364 |
| 35 | 0.014552 | -0.016706 | 0.001414 | -0.002242 | 0.002299 |
| 36 | 0.012225 | -0.009433 | 0.001344 | -0.002168 | 0.002170 |
| 37 | 0.009570 | -0.008320 | 0.001409 | -0.001875 | 0.002134 |
| 38 | 0.011110 | -0.009985 | 0.001517 | -0.001845 | 0.002202 |
| 39 | 0.008670 | -0.008905 | 0.001390 | -0.001571 | 0.001919 |
| 40 | 0.011047 | -0.007994 | 0.001285 | -0.001601 | 0.001889 |
| 41 | 0.012630 | -0.009923 | 0.001384 | -0.001736 | 0.002051 |
| 42 | 0.011490 | -0.011549 | 0.001350 | -0.001674 | 0.002018 |
| 43 | 0.012439 | -0.013953 | 0.001319 | -0.001723 | 0.002079 |
| 44 | 0.015883 | -0.017148 | 0.001433 | -0.002163 | 0.002568 |
| 45 | 0.011392 | -0.013071 | 0.001341 | -0.002393 | 0.002633 |
| 46 | 0.012741 | -0.011564 | 0.001454 | -0.002803 | 0.002881 |
| 47 | 0.012965 | -0.016353 | 0.001539 | -0.003147 | 0.003059 |
| 48 | 0.017026 | -0.017593 | 0.001850 | -0.003449 | 0.003329 |
| 49 | 0.018938 | -0.020511 | 0.002076 | -0.002774 | 0.003418 |
| 50 | 0.018105 | -0.021207 | 0.002076 | -0.003441 | 0.003736 |
| model | 0.025288 | -0.025327 | 0.002413 | -0.002256 | 0.003145 |

Continued on next page...

Table 5.7 – Continued from previous page...

| View | Max (m) | Min (m) | $+\mu$ (m) | $-\mu$ (m) | $\sigma$ (m) |
|---|---|---|---|---|---|
| median | 0.014552 | -0.013302 | 0.001812 | -0.002168 | 0.002568 |
| $\mu$ | 0.015551 | -0.014569 | 0.001931 | -0.002175 | 0.002580 |
| $\sigma$ | 0.004061 | 0.004747 | 0.000532 | 0.00048 | 0.000413 |

Table 5.7: Surface deviations of each individual view compared to the ground truth model. Positive values indicate a view that extends above the surface of the reference, and negative values indicate a view that extends below the surface of the reference. The row labeled "model" indicates the simultaneous evaluation of all views.

Examining Table 5.7 indicates that on average the surface error of any given view ranges from -2.2 mm to 1.9 mm. Equivalently, the mean surface error can be written as approximately -0.15 mm, with a standard deviation of 2.6 mm. However, there also appear to be outliers in multiple views as far as 2.5 cm away from the ground truth surface. The data presented here allows us to easily identify the worst perpetrators and take action to remove these outliers if desired. A full series of visualizations for this data is tabulated in Appendix D.1.

| | Absolute Errors | | | | |
|---|---|---|---|---|---|
| View | Max (m) | Min (m) | $+\mu$ (m) | $-\mu$ (m) | $\sigma$ (m) |
| 1 | 0.017947 | -0.017042 | 0.001141 | -0.001399 | 0.001654 |
| 2 | 0.017103 | -0.014050 | 0.001204 | -0.001447 | 0.001773 |
| 3 | 0.017236 | -0.014022 | 0.001214 | -0.001436 | 0.001765 |
| 4 | 0.017220 | -0.014111 | 0.001236 | -0.001425 | 0.001774 |
| 5 | 0.010910 | -0.014175 | 0.001239 | -0.001418 | 0.001791 |
| 6 | 0.022936 | -0.012635 | 0.001337 | -0.001429 | 0.001958 |
| 7 | 0.022949 | -0.012068 | 0.001359 | -0.001445 | 0.002002 |
| 8 | 0.024150 | -0.012476 | 0.001388 | -0.001455 | 0.002038 |
| 9 | 0.024121 | -0.012161 | 0.001384 | -0.001449 | 0.002037 |

Continued on next page...

Table 5.8 – Continued from previous page...

| **Group** | **Max (m)** | **Min (m)** | $+\mu$ **(m)** | $-\mu$ **(m)** | $\sigma$ **(m)** |
|---|---|---|---|---|---|
| 10 | 0.024027 | -0.016325 | 0.001431 | -0.001408 | 0.002045 |
| 11 | 0.023795 | -0.016368 | 0.001432 | -0.001406 | 0.002013 |
| 12 | 0.020298 | -0.016308 | 0.001434 | -0.001350 | 0.001967 |
| 13 | 0.020689 | -0.024973 | 0.001470 | -0.001290 | 0.001972 |
| 14 | 0.020390 | -0.025389 | 0.001465 | -0.001238 | 0.001935 |
| 15 | 0.016846 | -0.025330 | 0.001434 | -0.001212 | 0.001971 |
| 16 | 0.019874 | -0.025015 | 0.001368 | -0.001155 | 0.001900 |
| 17 | 0.024550 | -0.025371 | 0.001888 | -0.001726 | 0.003037 |
| 18 | 0.018162 | -0.025392 | 0.001267 | -0.001241 | 0.001906 |
| 19 | 0.018845 | -0.018536 | 0.001230 | -0.001210 | 0.001793 |
| 20 | 0.017233 | -0.017793 | 0.001169 | -0.001343 | 0.001825 |
| 21 | 0.016218 | -0.013149 | 0.001154 | -0.001428 | 0.001847 |
| 22 | 0.013892 | -0.010648 | 0.001087 | -0.001465 | 0.001792 |
| 23 | 0.023699 | -0.025253 | 0.001206 | -0.001657 | 0.002045 |
| 24 | 0.023699 | -0.025274 | 0.001299 | -0.001816 | 0.002090 |
| 25 | 0.012197 | -0.011853 | 0.001202 | -0.001820 | 0.001956 |
| 26 | 0.011619 | -0.012865 | 0.001093 | -0.001809 | 0.001882 |
| 27 | 0.011800 | -0.012689 | 0.001017 | -0.001784 | 0.001819 |
| 28 | 0.011471 | -0.025211 | 0.001705 | -0.002497 | 0.002693 |
| 29 | 0.025250 | -0.025198 | 0.001883 | -0.002357 | 0.002735 |
| 30 | 0.025187 | -0.022906 | 0.002038 | -0.002546 | 0.002891 |
| 31 | 0.025325 | -0.022562 | 0.002151 | -0.002619 | 0.003017 |
| 32 | 0.018648 | -0.016036 | 0.002256 | -0.002727 | 0.003127 |
| 33 | 0.018488 | -0.015885 | 0.002445 | -0.002827 | 0.003292 |
| 34 | 0.013730 | -0.014396 | 0.001364 | -0.000814 | 0.001440 |
| 35 | 0.013739 | -0.013293 | 0.001291 | -0.000801 | 0.001355 |
| 36 | 0.009946 | -0.009410 | 0.001261 | -0.000772 | 0.001299 |

Continued on next page...

Table 5.8 – Continued from previous page...

| Group | Max (m) | Min (m) | $+\mu$ (m) | $-\mu$ (m) | $\sigma$ (m) |
|---|---|---|---|---|---|
| 37 | 0.009946 | -0.012213 | 0.001259 | -0.000848 | 0.001377 |
| 38 | 0.011501 | -0.012213 | 0.001237 | -0.000842 | 0.001379 |
| 39 | 0.011480 | -0.009459 | 0.001290 | -0.000863 | 0.001381 |
| 40 | 0.010452 | -0.012869 | 0.001041 | -0.000982 | 0.001427 |
| 41 | 0.010547 | -0.012858 | 0.001023 | -0.001020 | 0.001440 |
| 42 | 0.011848 | -0.012897 | 0.001021 | -0.001301 | 0.001627 |
| 43 | 0.011826 | -0.012704 | 0.001038 | -0.001397 | 0.001688 |
| 44 | 0.012398 | -0.014780 | 0.001086 | -0.001664 | 0.001800 |
| 45 | 0.012705 | -0.014380 | 0.001085 | -0.001738 | 0.001760 |
| 46 | 0.012879 | -0.017495 | 0.001262 | -0.002310 | 0.002539 |
| 47 | 0.014836 | -0.013112 | 0.001147 | -0.002004 | 0.001900 |
| 48 | 0.017280 | -0.017735 | 0.001745 | -0.003133 | 0.003639 |
| 49 | 0.015704 | -0.018573 | 0.001548 | -0.003248 | 0.003432 |
| model | 0.025375 | -0.025399 | 0.001462 | -0.001557 | 0.002336 |
| median | 0.017227 | -0.014588 | 0.001279 | -0.001433 | 0.001903 |
| $\mu$ | 0.017259 | -0.016857 | 0.001376 | -0.001603 | 0.002043 |
| $\sigma$ | 0.005060 | 0.005177 | 0.000322 | 0.000598 | 0.000563 |

Table 5.8: Surface deviations of each group compared to the ground truth model. Positive values indicate that a group extends above the surface of the reference, and negative values indicate that a group extends below the surface of the reference. The row labeled "model" indicates the simultaneous evaluation of all groups.

Next, according to Table 5.8 the average surface error of any given group ranges anywhere from -1.6 mm to 1.4 mm, equating to a rough mean surface error of -0.1 mm, with a standard deviation of 2.0 mm. A comparison of these results to those of Table 5.7 validates our hypothesis that grouping similar views together produces superior alignment than using the point clouds individually. However, there seems to be very little change in the maximum and minimum surface deviations, which we continue to associate

with a very small number of points, based on the visualizations of this data in Appendix D.2.

Lastly, we remind the reader that the model we evaluated in this chapter is constructed using subsampled data, which blurs many of the finer details that are visible on the ground truth model; an effect that is easily observed in Figure 5.3. Optionally, restoring the full resolution of the data should further reduce the error in our reconstruction and improve the quality of the texture. Overall, the results presented in this section show that our system performs very well when working with data sets of this type. Based on our registration analysis, we feel that due to the amount of translational correction required to achieve convergence between many of the point sets, it is highly unlikely that registration could be achieved for this data set without our heuristics.

Overall, we feel that the acceleration data provided by the inertial sensor is not as accurate as expected, but the spatial orientation provided is invaluable as an initial condition for speeding up the registration process. Furthermore, we feel that that translation obtained from the INS could still be improved either through advanced filtering and integration methods, or through the use of a more sensitive accelerometer because the current one can measure accelerations far beyond anything our rig should experience during a typical acquisition. The appeal of including an INS in our 3D scanning system continues to be the initial coarse registration that facilitates rapid fine registration and minimizes superfluous processing, but there remains room for improvement.

## 5.4 Alternate Data Profile

Our second data set is a sequence 50 frames in length consisting of the bottom half of the totem pole found in Confederation Park in Ottawa. We chose this totem for our second data set because it is precisely the kind of subject we expect our system to be used to document; it is an immobile object of cultural interest, with visual appeal, and located in a public outdoor setting where using a bulkier system would be troublesome.

It also presents the opportunity to test our system on an object with a different set of surface characteristics than the artwork. The surface of the totem is mostly smooth to the touch, but its distinctive carvings provide ample depth information that can be utilized for stereo correspondence using our setup. This data set will be referred to as the "totem" sequence from now on; a few sample frames are illustrated in Figure 5.4 and the full 50 frame sequence is available in Appendix C.2.

For consistency, the totem sequence was captured at a resolution of $1024 \times 768$ pixels

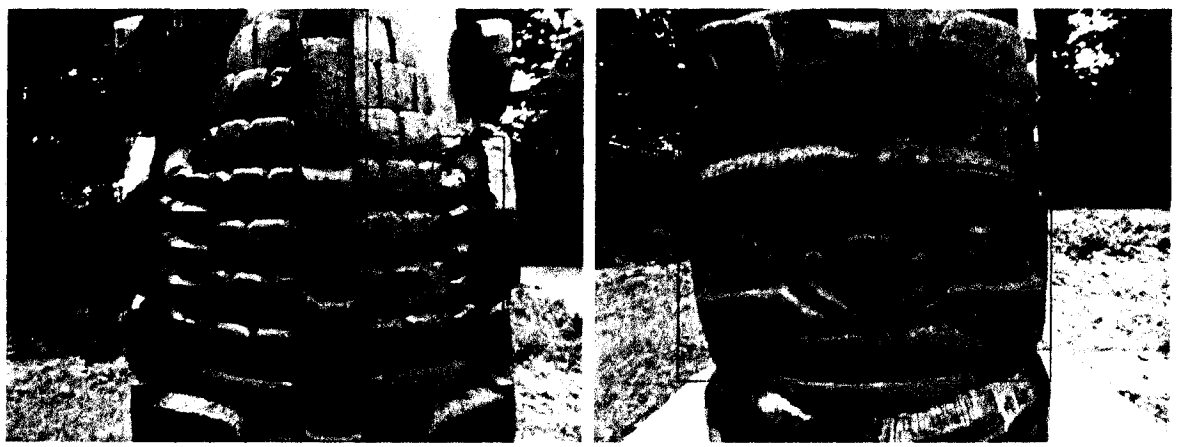
(a) (b)

Figure 5.4: Subject of the secondary "totem" data set as it appears (a) near the bottom, and (b) near the top.

using the same settings as the artwork data set. The data was subsampled by a factor of 100 for fast registration, but no subsampling was used prior to surface reconstruction and texturing in order to demonstrate the best possible visual representation of the subject that our system is capable of producing. Processing for this data set was performed using the same calibration matrix as the artwork data set because there was no change in the hardware configuration between the two. The calibration matrix is displayed in Equation 5.1.

Camera motion during acquisition was smooth and consistent, moving up and down along the surface of the subject from left to right. Specifically, the left side of the totem was recorded in an upward motion, the center in a downward motion, and the right side in an upward motion once more, while ensuring that there was always overlap between the three sections. We were unable to record the full height of the totem pole because it was simply too tall to do so without a lift. The disparity range indicated in Table 3.1 necessitated that the camera be held about half a meter to a meter away from the subject for the stereo correspondence API to function properly. This type of motion can be classified as mostly translational with minimal rotation, which is a mix of the types that we discussed in Section 4.6.

Unfortunately, we were unable to register the totem data set using our automated system. The results appeared to suffer from "sliding", a common but undesirable effect that can occur when using the iterative closest point method to align mostly planar sur-



(a) (b)

Figure 5.5: Regions of ICP failure for the totem data set (a) where features repeat along the direction of camera motion, and (b) where the relief is not sufficiently distinctive.

faces. Essentially, two planes with minimal or repetitive features tend to continue sliding over one another in an attempt to minimize the ICP error measure until a sufficiently substantial feature is found to "latch onto", or the ICP exceeds its iteration limit. In this case, the totem pole is basically a curved surface with relief that in a few locations is not substantial enough to be uniquely characterized by our ICP variant. We identified two such locations in the data set, which are displayed in Figure 5.5.

Although untested, we theorize that it may be possible for our automated system to succeed with this data set if care is taken to engineer a less translational acquisition path, such as a lateral spiral. This motion would produce more constraining features (*e.g.,* curvature along the path of motion) that ought to limit the amount of "sliding" that occurs, as well as a larger number of similarity groups than the vertical motion we used. Additionally, a greater number of groups generally produces more fast registrations involving fewer views, while the opposite is true for fewer groups.

Therefore, we attribute this problem to a failure in our ICP variant, but we feel that our overall procedure is still sound. To test this theory, we replace our ICP module with the proprietary implementation found in Geomagic, and execute our three-stage registration procedure manually, as described in Chapter 4. Although a small percentage of frames were rejected, ultimately this test confirmed our suspicions about the origin of the failure. The global solution produced by repeating the procedure manually is illustrated in Figure 5.6.

Figure 5.6: Model of the totem data set constructed using our procedure with an alternative third–party ICP module.

This result is very good overall, but it is immediately apparent that additional data is needed to fill the remaining holes. Lastly, we feel that if the camera motion were tracked more accurately by the INS, it would reduce the likelihood of encountering a failure in the ICP module, the result of which is dependent on good initial conditions. While the INS is not yet as accurate as we would like it to be, it still provides a sufficiently good estimate of motion in order for ICP (and our heuristic) to initialize processing. Reviewing the errors from both data sets brings to light some of the current limitations of our 3D scanning system, which we summarize in the next section.

## 5.5 System Limitations

Based on our analysis of the internal registration errors for the artwork data profile, we can conclude that one of the limiting factors of our system is the accuracy of our motion tracking over larger distances. The ultimate goal of our system and others like it is to determine the correct positions and orientations as accurately as possible from sensor measurements alone to eliminate any need for sophisticated post–processing techniques.

Unfortunately, it appears that our inertial sensor is not capable enough to meet this goal at this time, and despite our best efforts, its accelerometer readings likely still contain drift that accumulates throughout the sequence.

Secondly, our work with the totem data profile highlights a deficiency in our ICP variant that makes it susceptible to "sliding" when working with smooth or repetitive data. This occurs when a pair of point clouds continues to slide over one another trying to minimize the ICP error measure even if it is not the desired global solution because there are too few unique features to lock them together. This is a common problem associated with the iterative closest point algorithm, but we had hoped that our use of normal–space sampling and the compatibility of normals to determine point–to–point correspondence would have eliminated this problem. Testing with an alternative implementation from Geomagic shows that our heuristics are sound.

We also speculate that the direction of motion during acquisition of the totem sequence also impacted the effectiveness of our ICP module. Referring back to Figure 5.5(a), we see that the indicated regions repeat vertically in the direction of motion used to record this section of the sequence. Sliding another similar frame vertically across this one is unlikely to induce significant change in the alignment error measure, so no solution is found. On the other hand, if the camera motion had been lateral instead of vertical, the same regions would probably have been recognized as significant features, and convergence would have been achieved.

We conclude that our 3D scanning system is currently more capable at registering closed–loop "rotational" sequences, than it is at registering "translational" sequences as defined earlier. This is still a reasonable result if we consider that there is a large proportion of potential subjects that necessitate a closed–loop acquisition to produce a full 360° model. Most importantly, we believe that convergence could not have been achieved using only ICP for either of our data sets because of the large amount of translational error reported by our internal measurements, more so for the the totem data set than the artwork.

## 5.6 Summary

The results presented in this chapter show that our system is a viable solution for creating fully textured polygonal models of many outdoor objects for cultural documentation, industrial design, reverse engineering and prototyping, crime scene reconstruction and other similar applications. The model generated from our primary data set exhibited

good visual quality, with only minor blurring in the texture caused by a reduction in the number of samples used for surface reconstruction and texturing. A comparison of our reconstruction against a ground truth model acquired using laser triangulation also confirms that our camera–based scanner is accurate enough to reproduce good surface detail.

There were also certain precision and convergence issues in our system which need to be addressed. Even though our 3D model of the artwork is a good visual and geometric representation of the subject matter, quantitative analysis of the internal registration errors indicated higher than expected values for camera translation. Additionally, our registration module has trouble converging to a global solution when working with translational sequences, and flat or repetitive surfaces. We suspect that efforts to improve the filtering and integration of accelerometer data will alleviate some of the precision errors, and utilizing a more robust ICP variant has been confirmed to alleviate many of the convergence issues.

Also, it seems that the accelerometer we used may not be sensitive enough (*i.e.*, $\pm$ 50 $m/s^2$ is a large range) to accurately represent the relatively small motions made during the acquisition sequence. It is likely that a more sensitive accelerometer (*i.e.*, in the $\pm$ 2-10 $m/s^2$ range) would be far more capable at quantifying the motions experienced by our rig during a typical acquisition.

# Chapter 6

# Conclusion

In this thesis, we proposed a stereo–based outdoor acquisition system to be used in conjunction with the standard 3D scanning pipeline, whose instrumentation and registration techniques are sufficient enough so that no extra visual tracking is required to produce a model from multi–view data, if care is taken. We have assembled a test scanning system to realize this goal, as discussed in Section 1.4. Our approach eliminates the need for elaborate capture rigs or rigid motion, and our system may be adapted to a wide range of scanning conditions. We have outlined the steps for generating polygonal models from raw sensor data using our system, and addressed some of the problems which arose in our automated registration procedure during testing (*e.g.*, sensor drift, failure to converge). Results indicate that our thesis is correct for some acquisition scenarios while the large error in translation and rotation made others fail.

A key contribution arising from our work is our three–stage heuristic algorithm for multi view registration of surface point clouds which aims to maintain generality for scanning a variety of subjects with different characteristics. Our system also incorporates a technique to determine "similarity groupings" which lump together neighbouring point clouds based on the orientation of the camera to facilitate faster, more reliable convergence. Additionally, we condition the data from both sensors with filtering and digital processing to reduce background noise in the stereo data, and to reduce drift and high frequency noise in the acceleration data. Our ground truth comparison validates our hypothesis that grouping similar views together produces a superior reproduction of the subject than using the point clouds individually.

One of the limiting factors of our system is related to the accuracy of the sensor hardware over large distances. Despite our efforts, our internal measurements indicate

that the accelerometer readings still contain drift that accumulates throughout the sequence, thereby reducing the quality of our motion tracking and increasing the strain on the subsequent registration module. Auspiciously, this problem has limited impact on our system, affecting only those data sets that favour translational motion for their acquisition. This problem could be corrected by revising our signal processing techniques to focus on the long term stability of the sensor data, which could be implemented with little additional cost to our system.

Secondly, our work identified a deficiency in our ICP variant that makes it susceptible to "sliding" when working with smooth or repetitive data, resulting in a failure to converge. While this is a common problem associated with the iterative closest point algorithm, we had hoped that our use of normal-space sampling and the compatibility of normals for point-to-point correspondence would have eliminated this problem. Fortunately, testing with an alternative implementation showed that our heuristics are sound, and this issue can be resolved by utilizing a more robust and stable variant of ICP.

Applications for mobile scanning systems such as ours are available in many fields including cultural documentation, crime scene reconstruction, entertainment and more. While there is definitely room for improvement, based on the results presented in Chapter 5, we believe our system is a good option for reproducing visually and geometrically accurate 3D models of outdoor subject matter. Of note is the fact that without our heuristics and registration procedure, neither of the reconstructions presented therein would have been possible because it is highly unlikely that ICP alone is sufficient to align the raw data into its global solution.

## 6.1 Future Work

We would like to improve our scanning system while maintaining its ease of use and mobility. Our experiences using our scanning system in the field highlighted some of the areas where its usability could benefit from additional improvement. The current implementation requires a single user to initiate the acquisition sequence with a script that includes a lengthy delay to allow the user to place the host laptop into a backpack while maintaining all of the appropriate wired connections with the sensors. Introducing two-way wireless connectivity between the hand-held unit and the laptop has the potential to greatly increase the ease of use and efficiency of acquisition by eliminating the setup time that is currently required.

It would also be beneficial to include a viewfinder on the hand-held unit because there

is currently no way to know what part of a scene is being recorded without reviewing the resulting video. The user must perform the acquisition blindly and hope that their aim is good throughout the sequence. This means that often times the acquisition of consistent data becomes a trial and error affair, which is not particularly time efficient.

Compliance of our calibration interface could also be improved by eliminating the caveat for the Wavefront OBJ file that defines the geometry of our calibration target. Currently, the definition of each face must include all of the vertices that intersect the face as well. If not properly executed (*e.g.*, clockwise rotation instead of counterclockwise, appending vertices to a face arbitrarily), these additions could produce unforeseen artifacts when the file is loaded into the calibration interface. Calculating these intersections internally would eliminate this requirement at no additional cost. Alternatively, we could utilize a user extension to the PLY format to include this information and avoid these issues completely.

Possibly the most significant work yet to be done focuses on the filtering and processing of accelerometer measurements. Accelerometers are notoriously susceptible to drift in their measurements which is an undesirable quality in our system which relies on these measurements to produce an initial alignment of two point sets. Drift can come from many different sources, most commonly from changes in the temperature of the environment or the temperature of the device caused by extended use; a significant source of error in our system. Revising our signal processing techniques in Section 4.4 to utilize a Kalman filter to stabilize the signal from the accelerometer would be a advantageous. A Kalman filter is a recursive filter which estimates the state of a dynamic system when incomplete or noisy measurements are being used [35]. It is popular because of its simplicity, optimality, and robustness and would be ideal to use for estimating sensor motion.

Lastly, in Section 4.5 we described the variant of the ICP algorithm that we use to perfect the alignment of our multi view data. Unfortunately, this variant failed to meet all of our expectations. Results in Chapter 5 show that it performs very well when scanning "rotational" subjects, but fails to converge for those favouring "translational" camera motion. One of the design goals for our scanning system was for it to be applicable to a wide range of subjects. Consequently, we plan to incorporate a more robust and stable variant of the iterative closest point algorithm into our system. There are several options available to us to improve each element of our ICP module, including the addition of a correspondence weighting step. We would also like to explore the use of vertex colour information in our implementation.

# Appendix A

# Technical Sheets

The technical specifications of the Point Grey Bumblebee stereo camera are presented here [59]:

- 2 Sony ICX204 $\frac{1}{3}$ inch progressive scan colour CCDs.
- 640 × 480 square pixels at 30 Hz frame rate.
- 1024 × 768 square pixels at 15 Hz frame rate.
- 10 bit A/D.
- Shutter speed: $\frac{1}{8000}$ s to $\frac{1}{30}$ s @ 30 Hz at 640 × 480 or $\frac{1}{6000}$ s to $\frac{1}{15}$ s @ 15 Hz at 1024 × 768.
- Baseline: 12 cm.
- Choice of 2 mm, 4 mm or 6 mm focal length lenses (100°, 70°, and 50° HFOV respectively).
- Size: approximately 160 × 40 × 50 mm.
- Weight: approximately 375 g.
- Automatic/manual gain and shutter control with adjustable frame rate.
- Independent control of CCD gains for image intensity balancing.
- Color models provide Bayer tiled images for reconstruction on the host.

- One IEEE-1394a 6-pin connector.
- Power consumption: 2.1 W supplied by IEEE-1394a.

The technical specifications of the Xsens MTi AHRS are presented here [96]:

- Dynamic range: all angles in 3D.
- 0.05° angular resolution.
- $< 0.5°$ static accuracy (roll/pitch).
- $< 1°$ static accuracy (heading).
- 2° RMS dynamic accuracy.
- Full scale (standard): ±1200°/s rate of turn or ± 50 m/s$^2$ acceleration or ± 750 mGauss magnetic field.
- Linearity: 0.1% of FS rate of turn or 0.5% of FS acceleration or 0.2% of FS magnetic field.
- Bias stability (1 s): 5 °/s rate of turn or 0.02 m/s$^2$ acceleration or 0.5 mGauss magnetic field.
- Scale Factor stability (1 s): 0.05% acceleration or 0.5% magnetic field.
- Noise density: 0.1 °/s/vHz rate of turn or 0.02 ms/s$^2$/vHz acceleration or 0.5 mGauss (1 s).
- Alignment error: 0.1° rate of turn or 0.1° acceleration or 0.1° magnetic field.
- Bandwidth (standard): 40 Hz rate of turn or 40 Hz acceleration or 10 Hz magnetic field.
- Max update rate: 512 Hz for calibrated sensor data. 120 Hz for orientation data.
- RS-232 and USB (external converter) digital interface.
- Operating voltage: 4.5 - 30 V.
- Power consumption: 360 mW.
- Dimensions: 58 × 58 × 22 mm.

- Weight: 50 g.

Additionally, the MTi has the capacity to reset its coordinate system to one of the following four configurations:

**HEADING** Changes the global reference of North to the direction of the x–z plane of the MTi, also known as "bore–sighting". The output co-ordinate system is changed accordingly.

**GLOBAL** Changes both the global reference of North and Gravity (G), to the current sensor co-ordinate system (S).

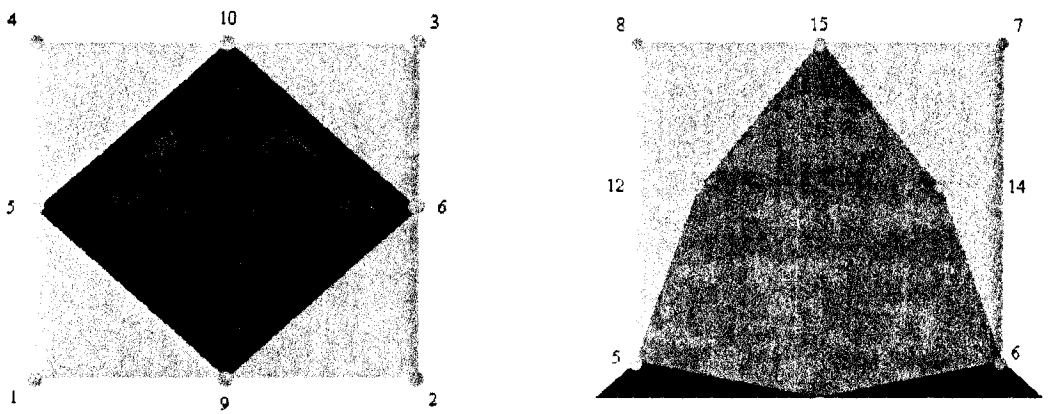
**OBJECT** Changes the output coordinate system (S) to the coordinate system of the object the MTi is strapped to (O). It is assumed the xz plane of the MTi is aligned with the xz plane of the object and the inclination (*i.e.*, attitude) is then changed to match O.

**ALIGN** Changes the output coordinate system by applying a subsequent combination of the object reset and a heading reset. This aligns the MTi output coordinate system to the object it is strapped on. The object should be pointed toward the desired new local North (GX) and the xz plane of the MTi should be aligned with the xz plane of the object.

# Appendix B

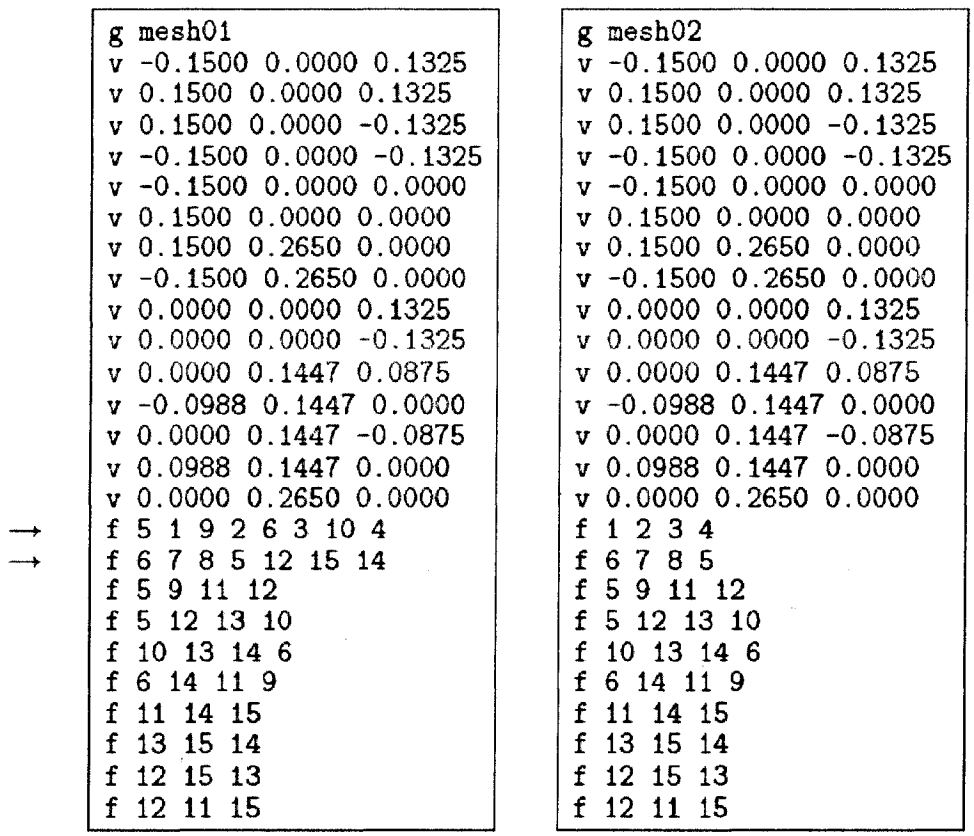
# Sample OBJ File

An example of the Wavefront OBJ file used for calibration of our 3D scanning system is displayed below on the left. A standard OBJ file is displayed below on the right, and contains only the vertex information needed to define the geometry of the calibration object. Examining the indicated rows in the left OBJ file shows additional vertices in those faces. These vertices are not required to define the geometry of the calibration object, but are listed as members of these faces because they are locations where edges intersect that face. Figure B.1 indicates these vertices on a 3D visualization of our calibration object.




(a) (b)

Figure B.1: Vertex information included to meet the caveat of the Wavefront OBJ file required for calibration.



```
g mesh01
v -0.1500 0.0000 0.1325
v 0.1500 0.0000 0.1325
v 0.1500 0.0000 -0.1325
v -0.1500 0.0000 -0.1325
v -0.1500 0.0000 0.0000
v 0.1500 0.0000 0.0000
v 0.1500 0.2650 0.0000
v -0.1500 0.2650 0.0000
v 0.0000 0.0000 0.1325
v 0.0000 0.0000 -0.1325
v 0.0000 0.1447 0.0875
v -0.0988 0.1447 0.0000
v 0.0000 0.1447 -0.0875
v 0.0988 0.1447 0.0000
v 0.0000 0.2650 0.0000
→ f 5 1 9 2 6 3 10 4
→ f 6 7 8 5 12 15 14
f 5 9 11 12
f 5 12 13 10
f 10 13 14 6
f 6 14 11 9
f 11 14 15
f 13 15 14
f 12 15 13
f 12 11 15
```

```
g mesh02
v -0.1500 0.0000 0.1325
v 0.1500 0.0000 0.1325
v 0.1500 0.0000 -0.1325
v -0.1500 0.0000 -0.1325
v -0.1500 0.0000 0.0000
v 0.1500 0.0000 0.0000
v 0.1500 0.2650 0.0000
v -0.1500 0.2650 0.0000
v 0.0000 0.0000 0.1325
v 0.0000 0.0000 -0.1325
v 0.0000 0.1447 0.0875
v -0.0988 0.1447 0.0000
v 0.0000 0.1447 -0.0875
v 0.0988 0.1447 0.0000
v 0.0000 0.2650 0.0000
f 1 2 3 4
f 6 7 8 5
f 5 9 11 12
f 5 12 13 10
f 10 13 14 6
f 6 14 11 9
f 11 14 15
f 13 15 14
f 12 15 13
f 12 11 15
```

Figure B.2: Comparison of the contents of the standard OBJ format (right), and the modified OBJ format for use in our system (left). Necessary alterations are indicated by the arrows on the left side, indicating the additional vertex indices required in the first two faces.

# Appendix C

# Data Sequences

## C.1 Artwork Sequence

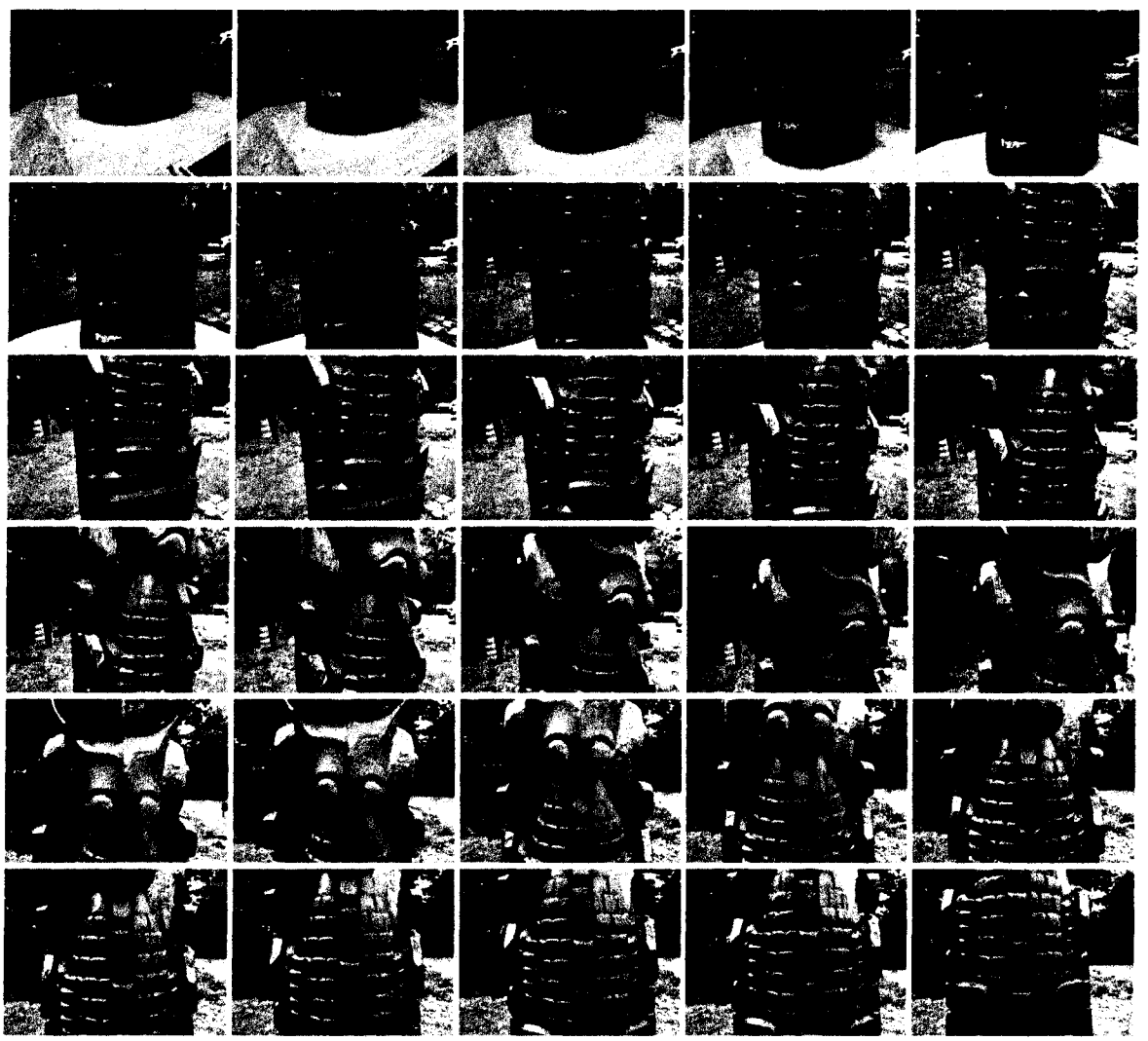
The full 50 frame sequence of the artwork data set featuring an abstract statue of an owl placed on a concrete path just outside the Colonel By building at the University of Ottawa. This sequence was captured at a resolution of $1024 \times 768$ pixels using the same settings as the calibration routine in Section 3.1. It was acquired in a single revolution using a smooth clockwise motion with the camera held about half a meter to a meter away

Continued on next page...

Continued from previous page...

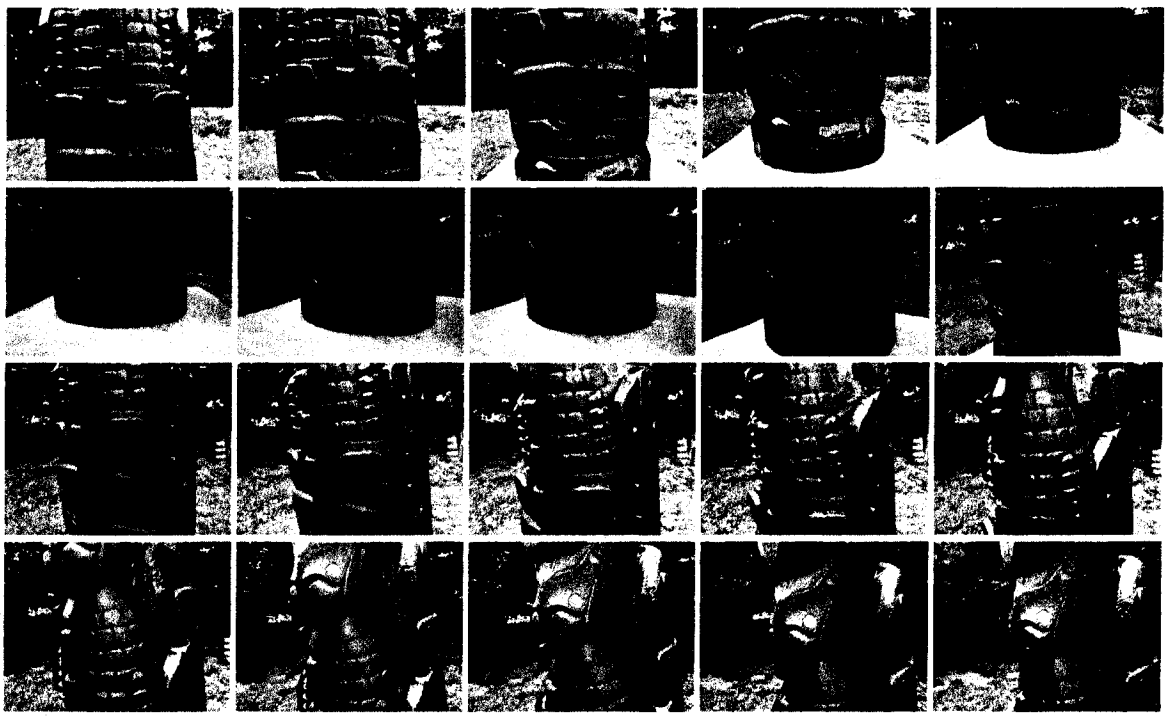
## C.2 Totem Sequence

The full 50 frame sequence of the totem data set featuring the bottom half of the totem pole found in Confederation Park in Ottawa. It was recorded at a resolution of $1024 \times 768$ pixels using the settings discussed in Section 3.1. Camera motion during acquisition was smooth and consistent, moving up and down along the surface of the subject from left to right, while ensuring that there was always overlap between the three sections. We were unable to record the full height of the totem pole because it was simply too tall to do so without a lift. The camera be held approximately half a meter to a meter away from the subject.



Continued on next page...

Continued from previous page...

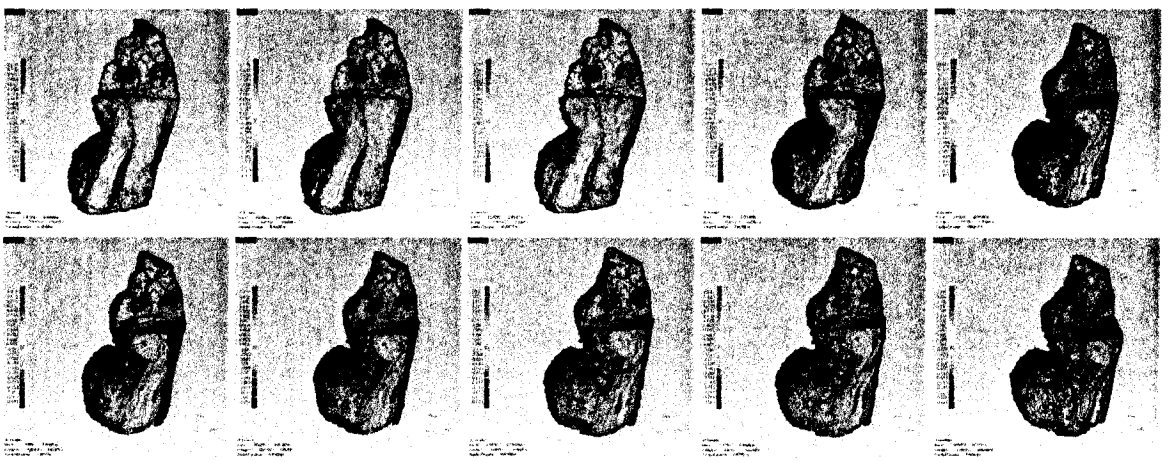


# Appendix D

# Ground Truth Visualizations

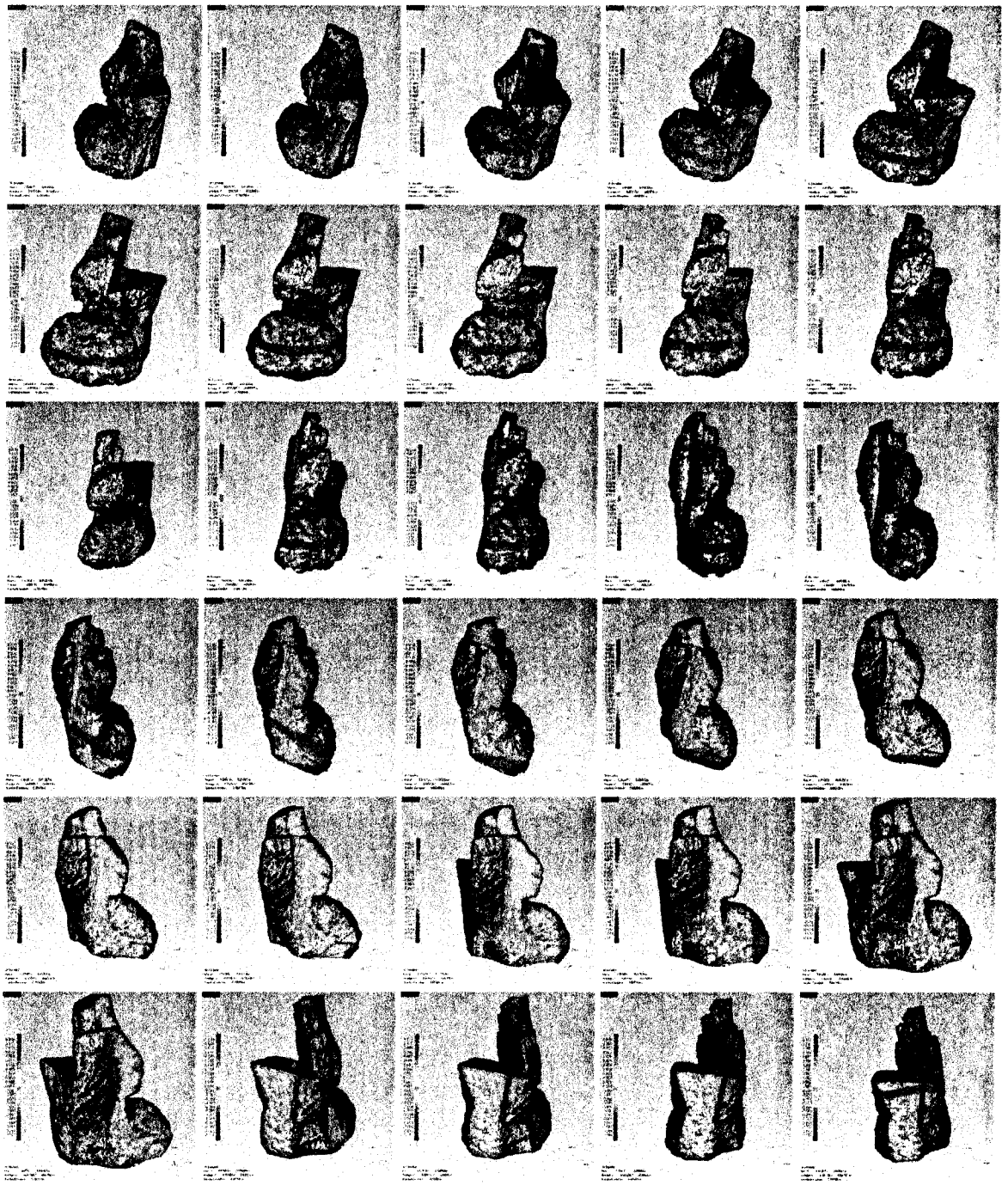
## D.1 View Visualizations

Colour coded visualizations of the surface deviation for the individual views from the artwork data sequence compared to the ground truth model using a point–to–surface distance measure. These minimum, maximum, mean and standard deviations of these results are tabulated in Table 5.7. Warm colours (i.e., oranges and reds) indicate locations where our model extends above the surface of the ground truth model, and cold colours (i.e., teals and blues) indicate locations where it extends below the surface of the ground truth model. Green and yellow indicate regions of low surface deviation.



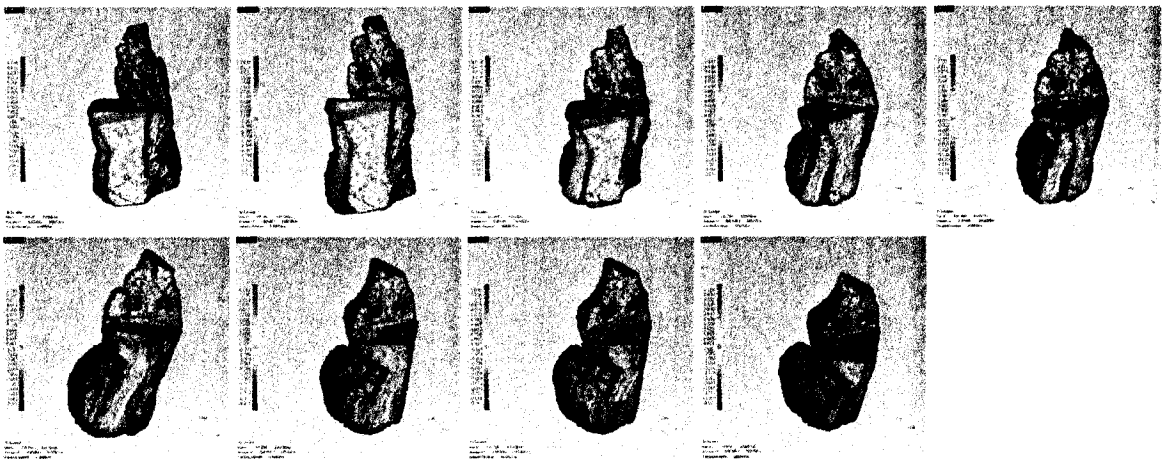
Continued on next page...

Continued from previous page...



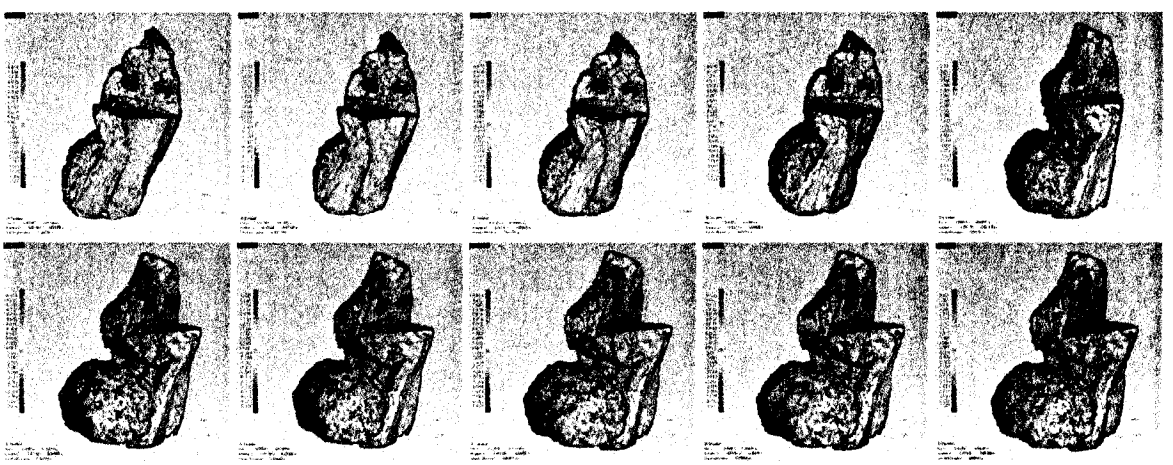
Continued on next page...

Continued from previous page...



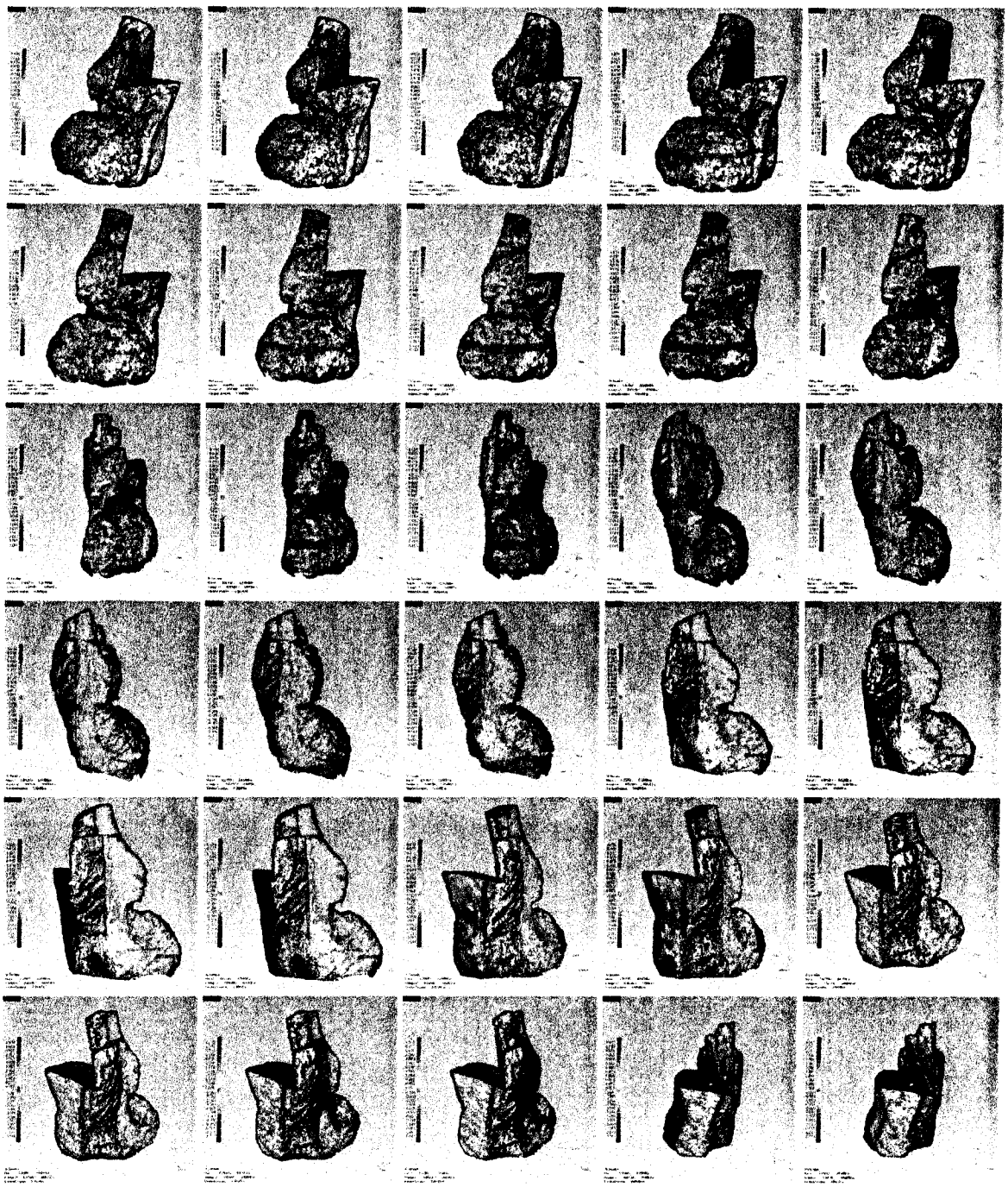
## D.2 Group Visualizations

Colour coded visualizations of the surface deviation for the groups generated for the artwork data sequence compared to the ground truth model determined using a point–to–surface distance measure. These minimum, maximum, mean and standard deviations of these results are tabulated in Table 5.8. Warm colours (i.e., oranges and reds) indicate locations where our model extends above the surface of the ground truth model, and cold colours (i.e., teals and blues) indicate locations where it extends below the surface of the ground truth model. Green and yellow indicate regions of low surface deviation.



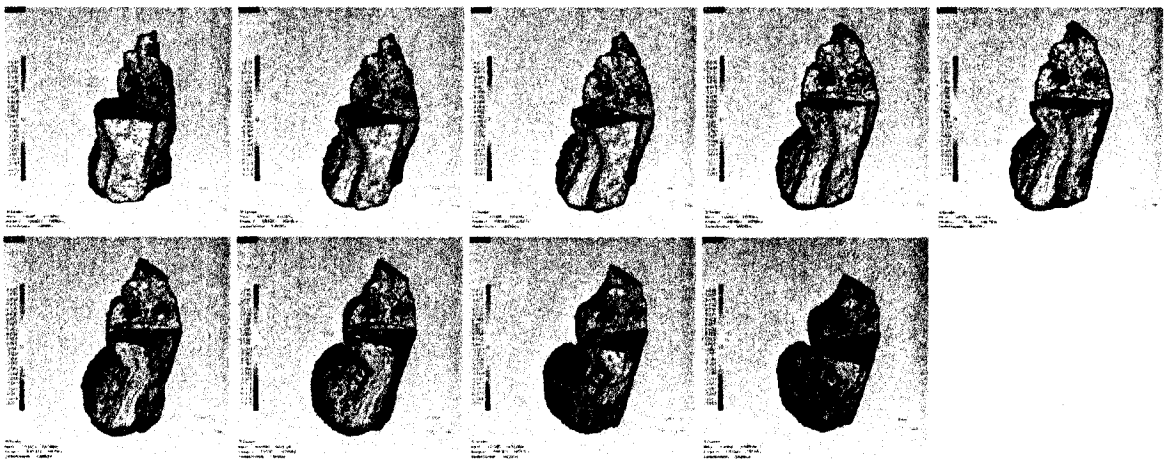
Continued on next page...

Continued from previous page...



Continued on next page...

Continued from previous page...



# Bibliography


[1] P. Aggarwal, Z. Syed, and N. El–Sheimy. Thermal calibration of low cost MEMS sensors for land vehicle navigation system. In *Proceedings of the IEEE 68th Vehicular Technology Conference (VTC)*, volume 11, pages 2859–2863, May 2008.

[2] P. Aggarwal, Z. Syed, X. Niu, and N. El–Sheimy. A standard testing and calibration procedure for low cost MEMS inertial sensors and units. *The Journal of Navigation*, 61(2):323–336, 2008.

[3] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3), 1989.

[4] H. H. Baker. Edge based stereo correlation. In L. S. Baumann, editor, *Proceedings of the 1980 ARPA Image Understanding Workshop*. Science Applications International Corporation, April 1980.

[5] S. T. Barnard. Stochastic stereo matching over scale. *International Journal of Computer Vision*, 3(1), 1989.

[6] F. Bernardini and H. Rushmeier. The 3D model acquisition pipeline. *Computer Graphics Forum*, 21(2), June 2002.

[7] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cludio Silva, and Gabriel Taubin. The ball–pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[8] P. Besl and N. McKay. A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2), 1992.

[9] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *Proceedings of the 4th IEEE International Conference on Computer Vision*, May 1993.

[10] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1), 1996.

[11] A. F. Bobick and S. S. Intille. Large occlusion stereo. *International Journal on Computer Vision*, 33(3), 1999.

[12] R. C. Bolles, H. H. Baker, , and M. J. Hannah. The JISCT stereo evaluation. In *Proceedings of the 1993 DARPA Image Understanding Workshop (IUW)*, volume 93, April 1993.

[13] C. Bonivento. VIsual DEcoder by Touch (VIDET), 2000. Retrieved November 1st, 2008 from the World Wide Web: http://www-lar.deis.unibo.it/activities/videt.

[14] Y. Boykov, O. Veksler, and R. Zabih. A variable window approach to early vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(12), 1998.

[15] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23(11), 2001.

[16] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *Proceedings of the 1991 IEEE Conference on Robotics and Automation (ICRA)*, volume 3, April 1991.

[17] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Image and Vision Computing*, 10(3), 1992.

[18] J. C. K. Chou and M. Kamel. Quaternions approach to solve the kinematic equation of rotation of a sensor-mounted robotic manipulator. In *Proceedings of 1988 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, April 1988.

[19] J.J Craig. *Introduction to Robotics: Mechanics and Control*. Addison Wesley, 1986.

[20] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. August 1996.

[21] K. Daniilidis. Hand eye calibration using dual quaternions. *International Journal of Robotics Research*, 18(3), 1999.

[22] S. El-Etriby, A. Al-Hamadi, and B. Michaelis. Dense stereo correspondence with slanted surface using phase based algorithm. In *Proceedings of the 2007 IEEE International Symposium on Industrial Electronics (ISIE)*, June 2007.

[23] O. D. Fangeras and M. Hébert. The representation, recognition, and locating of 3D objects. *International Journal of Robotics Research*, 5(3), 1986.

[24] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. In *Proceedings on the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 261–268, June 2004.

[25] M. A. Fischler and R. C. Bolles. *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, pages 726–740. Readings in computer vision: issues, problems, principles, and paradigms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[26] P. Fua. Combining stereo and monocular information to compute dense depth maps that reserve depth discontinuities. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, August 1991.

[27] M.C. Gambino, R. Fontana, M. Greco, E. Pampaloni, L. Pezzati, P. Pingi, P. Cignoni, and R. Scopigno. Supporting the restoration of the Minerva of Arezzo. In *Proceedings of the 5th International Conference on Lasers in the Conservation of Artworks (LACONA)*, September 2003.

[28] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 6(6), 1984.

[29] Geomagic. http://www.geomagic.com.

[30] W. R. Hamilton. *Lectures on Quaternions*. Hodges and Smith, Dublin, 1853.

[31] M. J. Hannah. *Computer matching of areas in stereo images*. PhD thesis, Stanford University, 1974.

[32] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2003.

[33] R. Horaud and F. Dornaika. Hand–eye calibration. *International Journal of Robotics Research*, 14(3):195–210, 1995.

[34] Berthold K. P. Horn. Closed–form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, April 1987.

[35] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering*, 82(D):35–45, 1960.

[36] T. Kanade. Development of a video–rate stereo machine. In *Proceedings of 1994 ARPA Image Understanding Workshop (IUW)*, November 1994.

[37] K. Kanatani. Analysis of 3-d rotation fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 16(5):543–549, 1994.

[38] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *Proceedings on the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, December 2001.

[39] A. Kelly. Modern inertial and satellite navigation systems. Technical Report CMU-RI-TR-94-15, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 1994.

[40] S. J. Kim, D. Gallup, J.M. Frahm, A. Akbarzadeh, Q. Yang, R. Yang, D. Nistér, and M. Pollefeys. Gain adaptive real-time stereo streaming. In *Proceedings of the 5th International Conference on Vision Systems (ICVS)*, March 2007.

[41] A. Klaus, M. Sormann, and K. Karner. Segment–based stereo matching using belief propagation and a self–adapting dissimilarity measure. In *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*, volume 3, August 2006.

[42] M. Labrie and P. Hébert. Efficient camera motion and 3D recovery using an inertial sensor. In *Proceedings of the 4th Canadian Conference on Computer and Robot Vision (CRV)*, pages 55–62, May 2007.

[43] R.K. Lenz and R.Y. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3D machine vision metrology. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 10(5):713–720, 1988.

[44] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital Michelangelo project: 3D scanning of large statues. In *Proceedings of the 27th international Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 131–144, July 2000.

[45] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th International Conference on Computer Vision (ICCV)*, September 1999.

[46] D. Marr. *Vision*. W. H. Freeman and Company, New York, NY, USA, 1982.

[47] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262), 1976.

[48] J. Marroquin, S. Mitter, and T. Poggio. Probabilistic solution of ill-posed problems in computational vision. *Journal of the American Statistical Association*, 82(397), 1987.

[49] S. Mattoccia, F. Tombari, and L. Di Stefano. Stereo vision enabling precise border localization within a scanline optimization framework. In *Proceedings of the 8th Asian Conference on Computer Vision (ACCV)*, volume 2, November 2007.

[50] Meshlab. http://meshlab.sourceforge.net.

[51] P. Mordohai and G. Medioni. Stereo using monocular cues within the tensor voting framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(6), 2006.

[52] H. K. Nishihara. Practical real-time imaging stereo matcher. *Optical Engineering*, 23(5), 1984.

[53] T. Okatani and K. Deguchi. Robust estimation of camera translation between two images using a camera with a 3D orientation sensor. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR)*, August 2002.

[54] G. Olague, F. Fernández, C. Pérez, and E. Lutton. The infection algorithm: an artificial epidemic approach for dense stereo correspondence. *Artificial Life*, 12(4), 2006.

[55] S. Paquette. 3d scanning in apparel design and human engineering. *IEEE Computer Graphics and Applications*, 16(5):11–15, 1996.

[56] M. Petrov, A. Talapov, T. Robertson, A. Lebedev, A. Zhilyaev, and L. Polonskiy. Optical 3D digitizers: Bringing life to the virtual world. *IEEE Computer Graphics and Applications*, 18(3):28–37, 1998.

[57] L. Pezatti and R. Fontana. 3D scanning of artworks. COST G7, Brussels, Belgium, 2006.

[58] M. Pires and C. Borg. 3D laser scanning of architectural sites. COST G7, Brussels, Belgium, 2006.

[59] Point Grey Research. http://www.ptgrey.com.

[60] M. Pollefeys, D. Nistér, J. M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewénius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[61] W.H. Press, B.P. Flannery, W.T. Vetterling, and S.A. Teukolsky. *Numerical Recipes in C*. Cambridge University Press, 2 edition, 1992.

[62] K. Pulli. *Surface Reconstruction and Display from Range and Color Data*. PhD thesis, University of Washington, 1997.

[63] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proceedings of the 9th International Conference on Computer Vision (ICCV)*, volume 1, October 2003.

[64] A. Rodríguez, J.M. Espadero, D. López, and L. Pastor. Delaunay surface reconstruction from scattered points. In *Proceedings of the 2000 International Conference on Discrete Geometry for Computer Imagery*, volume 1953, 2000.

[65] G. Roth and E. Wibowo. An efficient volumetric method for building closed triangular meshes from 3D image and point data. In *Proceedings of the 1997 Graphics Interface (GI)*, May 1997.

[66] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings of the 3rd International Conference on 3D Digital Imaging and Modeling (3DIM)*, May 2001.

[67] T. W. Ryan, R. T. Gray, , and B. R. Hunt. Prediction of correlation errors in stereo-pair images. *Optical Engineering*, 19(3), 1980.

[68] D. Scharstein. Matching images by comparing their gradient fields. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, volume 1, October 1994.

[69] D. Scharstein. View synthesis using stereo vision. In *Lecture Notes in Computer Science (LNCS)*, volume 1583. Springer-Verlag, 1999.

[70] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. *International Journal of Computer Vision*, 28(2):155–174, 1998.

[71] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, 2002.

[72] D. Scharstein and R. Szeliski. Middlebury stereo vision page, August 2008. Retrieved October 31st, 2008 from the World Wide Web: http://vision.middlebury.edu/stereo.

[73] S. Se and P. Jasiobedzki. Instant scene modeler for crime scene reconstruction. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, page 123, 2005.

[74] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotics Research*, 21(8), August 2002.

[75] P. Seitz. Using local orientation information as image primitive for robust object recognition. In *Proceedings of the 1989 Society of Photo Optical Instrumentation Engineers (SPIE) Visual Communications and Image Processing IV*, volume 1199, November 1989.

[76] J. Shah. "a nonlinear diffusion model for discontinuous disparity and half-occlusion in stereo". In *Proceedings of the 1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 1993.

[77] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distributions of optical flow. In *Proceedings of the 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 1991.

[78] G. Y. Sirat, F. Paz, G. Agronik, and K. Wilner. Conoscopic holography. In *Advanced Topics in Optoelectronics, Microelectronics, and Nanotechnologies II*, volume 5972 of *Proceedings of the 2005 Society of Photo–Optical Instrumentation Engineers (SPIE) Conference*, pages 11–16, August 2005.

[79] L. Di Stefano and S. Mattoccia. Real–time stereo within the VIDET project. *Real-Time Imaging*, 8(5), 2002.

[80] G. W. Stewart. *Introduction to Matrix Computations*. Academic Press, 1973.

[81] K. H. Strobl and G. Hirzinger. Optimal hand–eye calibration. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2006.

[82] R. Szeliski and G. Hinton. Solving random–dot stereograms using the heat equation. In *Proceedings of the 1985 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 1985.

[83] Y. Taguchi, B. Wilburn, and L. Zitnick. Stereo reconstruction with mixed pixels using adaptive over–segmentation. In *Proceedings on the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, page 8, June 2008.

[84] F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda. Near real–time stereo based on effective cost aggregation. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR)*, December 2008.

[85] P. Treleaven and J. Wells. 3D body scanning and healthcare applications. *Computer*, 40(7):28–34, 2007.

[86] R. Tsai. A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4), 1987.

[87] R. Tsai and R. Lenz. Review of the two stage camera calibration technique plus some new implementation tips and new techniques for center and scale calibration.

In *Proceedings of the 2nd Topical Meeting on Machine Vision, Optical Society of America*. March 1987.

[88] R.Y. Tsai and R.K.Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Journal of Robotics and Automation*, 5(3), June 1989.

[89] A. van den Hengel, A. Dick, T. Thormählen, B. Ward, and P. H. S. Torr. VideoTrace: rapid interactive scene modeling from video. In *Proceedings of the 34th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, page 86, August 2007.

[90] L. Verlet. Computer experiments on classical fluids. *Phys. Rev.*, 159(1), June 1967.

[91] J. Vollmer, R. Mencl, and H. Muller. Improved Laplacian smoothing of noisy surface meshes. *Computer Graphics Forum*, 18(3):131–138, 1999.

[92] D. Wang, O. Hassan, K. Morgan, and N. Weatherill. Efficient surface reconstruction from contours based on two-dimensional delaunay triangulation. *International Journal for Numerical Methods in Engineering*, 65(5):734–751, 2006.

[93] Z. Wang and Z. Zheng. A region based stereo matching algorithm using cooperative optimization. In *Proceedings on the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, page 8, June 2008.

[94] Y. Weiss and W.T. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47(2), 2001.

[95] A. Willis, J. Speicher, and D. B. Cooper. Rapid prototyping 3D objects from scanned measurement data. *Image and Vision Computing*, 25(7):1174–1184, 2007.

[96] Xsens Motion Technologies. http://www.xsens.com.

[97] L. Xu and J. Jia. Stereo matching: an outlier confidence approach. In *Proceedings of the 2008 European Conference on Computer Vision (ECCV)*, volume 4, October 2008.

[98] Q. Yang, L. Wang, R. Yang, H. Stewénius, and D. Nistér. Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling.

*IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(3), 2009.