1-1-2014

Registration and Recognition in 3D

Alexander Evans Patterson IV
University of Pennsylvania, xandey@gmail.com

Registration and Recognition in 3D

Abstract
The simplest Computer Vision algorithm can tell you what color it sees when you point it at an object, but asking that computer what it is looking at is a much harder problem. Camera and LiDAR (Light Detection And Ranging) sensors generally provide streams pixel of values and sophisticated algorithms must be engineered to recognize objects or the environment. There has been significant effort expended by the computer vision community on recognizing objects in color images; however, LiDAR sensors, which sense depth values for pixels instead of color, have been studied less. Recently we have seen a renewed interest in depth data with the democratization provided by consumer depth cameras. Detecting objects in depth data is more challenging in some ways because of the lack of texture and increased complexity of processing unordered point sets. We present three systems that contribute to solving the object recognition problem from the LiDAR perspective. They are: calibration, registration, and object recognition systems. We propose a novel calibration system that works with both line and raster based LiDAR sensors, and calibrates them with respect to image cameras. Our system can be extended to calibrate LiDAR sensors that do not give intensity information. We demonstrate a novel system that produces registrations between different LiDAR scans by transforming the input point cloud into a Constellation Extended Gaussian Image (CEGI) and then uses this CEGI to estimate the rotational alignment of the scans independently. Finally we present a method for object recognition which uses local (Spin Images) and global (CEGI) information to recognize cars in a large urban dataset. We present real world results from these three systems. Compelling experiments show that object recognition systems can gain much information using only 3D geometry. There are many object recognition and navigation algorithms that work on images; the work we propose in this thesis is more complimentary to those image based methods than competitive. This is an important step along the way to more intelligent robots.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Computer and Information Science

First Advisor
Kostas Daniilidis

Keywords
calibration, depth image, object detection, point cloud, registration

Subject Categories
Computer Sciences

REGISTRATION AND RECOGNITION IN 3D

Alexander Patterson IV

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Doctor of Philosophy

2014

Supervisor of Dissertation

_____

Kostas Daniilidis, Professor of Computer and Information Science

Graduate Group Chairperson

_____

Val Tannen, Professor of Computer and Information Science

Dissertation Committee

Camillo J. Taylor, Professor of Computer and Information Science

Jianbo Shi, Professor of Computer and Information Science

Jean Gallier, Professor of Computer and Information Science

Philippos Mordohai, Assistant Professor of Computer Science

REGISTRATION AND RECOGNITION IN 3D

ACKNOWLEDGMENT

First I would like to thank my advisor Prof. Kostas Daniilidis for his continued support, encouragement, knowledge, and all of his contributions, this thesis would not have been possible without him. I would like to thank the members of my Dissertation Committee Prof. C.J. Taylor, Prof. Jianbo Shi, and Prof. Jean Gallier for serving on my dissertation committee. I also acknowledge Prof. Philippos Mordohai who generously agreed to serve on my committee and with whom I shared many great conversations over the course of our collaboration. I would also like to thank the late Prof. Ben Taskar; I learned much about machine learning and point cloud processing from our collaboration on the URGENT project.

I would like to thank my co-authors Ameesh Makadia, Xenophon Zabulis, and Oleg Naroditsky who significantly contributed to both my research and my understanding. I would also like to thank Jean-Philippe Tardif, Yanis Pavlidis, Alexander Toshev, and Kosta Derpanis for the many fruitful research discussions.

The GRASP lab has been a source of inspiration, and I would be lucky to stay in contact with the many friends I've made there. Goran Lynch, Mike Zargham, Ankita Kumar, Roy Anati, Matthieu Lecce, Menlong Zhu, Jason Owens, Cody Phillips, Rodrigo Carceroni, Nikhil Kelshikar, João Barreto, Timothee Cour, Mirko Visionati, Katerina Fraghiadaki, Weiyu Zhang, Gang Song, Jeff Byrne, Anthony Cowley, Babak Shirmohammadi, and Elena Bernardis I want to thank you for your friendship and inspiration.

I would also like to thank my parents Alex and Carol, my brother Daniel, and Katie Williams for their unwavering support and love throughout my graduate studies.

ABSTRACT

REGISTRATION AND RECOGNITION IN 3D

Alexander Patterson IV

Kostas Daniilidis

The simplest Computer Vision algorithm can tell you what color it sees when you point it at an object, but asking that computer what it is looking at is a much harder problem. Camera and LiDAR (Light Detection And Ranging) sensors generally provide streams pixel of values and sophisticated algorithms must be engineered to recognize objects or the environment. There has been significant effort expended by the computer vision community on recognizing objects in color images; however, LiDAR sensors, which sense depth values for pixels instead of color, have been studied less. Recently we have seen a renewed interest in depth data with the democratization provided by consumer depth cameras. Detecting objects in depth data is more challenging in some ways because of the lack of texture and increased complexity of processing unordered point sets. We present three systems that contribute to solving the object recognition problem from the LiDAR perspective. They are: calibration, registration, and object recognition systems. We propose a novel calibration system that works with both line and raster based LiDAR sensors, and calibrates them with respect to image cameras. Our system can be extended to calibrate LiDAR sensors that do not give intensity information. We demonstrate a novel system that produces registrations between different LiDAR scans by transforming the input point cloud into a Constellation Extended Gaussian Image (CEGI) and then uses this CEGI to estimate the rotational alignment of the scans independently. Finally we present a method for object recognition which uses local (Spin Images) and global (CEGI) information to recognize cars in a large urban dataset. We present real world results from these three systems. Compelling experiments show that object recognition systems can gain much information using

only 3D geometry. There are many object recognition and navigation algorithms that work on images; the work we propose in this thesis is more complimentary to those image based methods than competitive. This is an important step along the way to more intelligent robots.

Contents

List of Figures

Chapter 1

Introduction

Object recognition is one of the most basic problems in computer vision. In this thesis we will explore some topics with the ultimate goal of how to recognize objects within the range image sensing modality. LIght Detection And Ranging (LiDAR) is a type of sensing which senses pixels much like a camera, but unlike a camera it also measures the range to the object being imaged. There are three main methods for sensing depth: measuring the time of flight that light travels before returning to the sensor, projecting a structured pattern onto the scene and measuring the deformation of that pattern relative to an offset camera, and using two camera imagers and measurnig the deformation of the image, which is also known as stereo vision.

With the decreasing cost and increasing availability of LiDAR scanners and commercial depth cameras, the Microsoft Kinect cost $150 at its debut in 2010, there will be a greater need for powerful algorithms that can use this richer data source. Because of modern graphics cards, improvements in Stereo Vision [Scharstein and Szeliski, 2002], and cheaper and better LiDAR sensors, 3D data is becoming easier and faster to acquire. It behooves the vision community to use this extra information to move beyond the simple camera. With these new data sources comes the problem of how to use geometry in the most effective

Figure 1.1: Three laser scanners are shown. Clockwise from top left: the 3rdTech Deltasphere, the Hokuyo UTM-30LX, and the Microsoft Kinect sensor.

way possible. Range alignment may not outperform state of the art bundle adjustment [Lourakis and Argyros, 2009] (a particularly effective pose esimation algorithm for use on camera imagry) in a head to head competition, but it provides a different source of pose estimation that can augment estimates from other sensors. Object detection using geometry is not solved with cameras either, but at the very least range information will allow objects with different textures to be classified independently of those textures. This is why we feel that these algorithms will help robots and systems to understand the world around them and make better and more complicated tasks available to automation in the future.

1.1 3D Sensors

For this thesis we will focus on data acquired by 3D depth sensors. The most basic unit of measurement will therefore be a range point. These points can be sensed using a few different methods. In this section we will explain those methods and what strengths and challenges are present for each.

1.1.1 Time of Flight Sensors

The most direct form of depth sensors is one that uses the time of flight method. These sensors emit light in one direction at a time and measure the distance that light has traveled before returning. The distance can be measured either by starting a clock when the light pulse is emitted and counting the elapsed time down to sub-nanosecond until it returns, or by modulating the light at a known frequency and measuring the interference in the returned and emitted light. Both of these methods produce a reliable direct depth measurement using a laser beam.

Once a reliable depth measurement is produced the sensor proceeds to sweep the measuring beam over the scene. The sweeping usually happens either by mounting the sensor on a spinning module like the Velodyne sensor, or by rotating a mirror to sweep the beam over the scene like a SICK sensor (an illustration of 2D line scan LiDAR's are shown for the Hokuyo and SICK sensors in figures 1.3 and 1.2).

The Deltasphere scanner pictured in figure 1.1 combines these two methods to capture a spherical scan by having both a rotating mirror and a rotation stage that moves the entire scanner perpendicular to the mirror.

One problem that frequently crops up with line scan LiDAR scanners is that the laser line may not be very narrow so it will actually capture the closest point to the scanner within the enlarged cone shaped region of influence. Another problem is that the measurement may contain depth bias. For instance, we noticed that dark regions with the Hokuyo UTM-30LX tended to have a different depth bias from light regions. In images, this concept is not important because all of the measurements happen on an imaging plane and the fact that the pixels don't overlap is enough to guarantee they won't affect each other. The effect of having too wide a beam angle can be compared to an out of focus camera, but instead of linearly mixing the intensity values from neighboring pixels, there is a winner take all condition where the closest point is the one that gets returned. With LiDAR this effect will

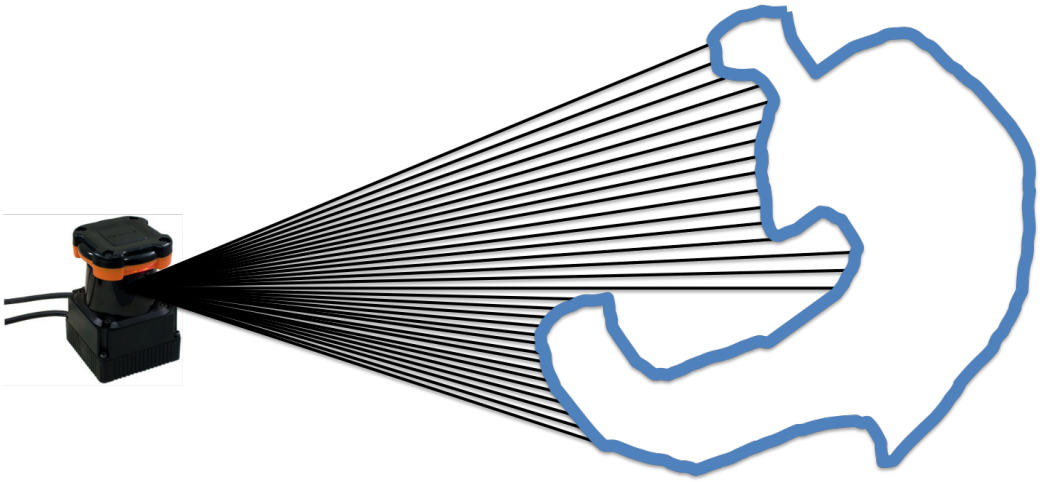

Figure 1.2: An illustration showing the scan lines produced by the Hokuyo 2D Line scanner. At each time step the internal mirror moves 0.25° and scans another ray. Notice the density is higher closer to the scanner, and the occlusion pattern that is produced.

make things more difficult because, firstly a surface is often scanned from two locations which means that the surface will have different spatial sampling rates depending on the angle of incidence of the scan and the distance to the scanner. Secondly, some scanners have a 2D line scanner that is moved throughout the scene, so in this case the scanner has one point density in the direction of the scan plane and a different point density when measured perpendicular to the scan plane. This scanning method is known as push-broom because the scanner covers an area somewhat like a push-broom sweeping a floor. These features of LiDAR scans serve to confound many range scan algorithms, and usually must be accounted for in the data processing somewhere.

1.1.2 Triangulation Sensors

The triangulation method of scanning uses two known imaging locations. That is to say two locations where we know the distance between them and the relative orientations of the sensors. The distance between the sensors is known as the baseline. The combined

Figure 1.3: The left image is an illustration of how the laser scans for line scan LiDAR scanners, a mirror deflects the laser beam in the desired direction. The central image shows an actual sick scanner which works the same way as the illustration. And the right image shows the Velodyne Laser scanner which rotates the entire laser assembly and sweeps a number of laser scanning lines across the image.

sensor must determine correspondences between the two images produced and then from the correspondence it uses triangulation to compute the full 3D point location.

In general, the problem of determining depth from two imaging locations can be computed from the well known equations for depth estimation from calibrated cameras:

$$z = \frac{bf}{x_l - x_r}, x = \frac{x_l z}{f}, y = \frac{y_l z}{f}.$$

Here the vector $[x, y, z]^\top$ is the estimated 3D point, x_l and x_r are the x location of the point in the left and right images respectively, f is the focal length of the camera, and b is the baseline. More details on stereo geometry can be found in standard machine vision reference books such as [Hartley and Zisserman, 2000].

The main challenge for the capturing device is how to best determine the correspondence between sensed points. We break up the scanners into two groups based on the method used. The first is structured light scanners which project a known pattern from the first location, and then measure this pattern's deformation from the second location. The second method is to use passive stereo cameras which use image properties such as color and intensity to determine the correspondence in the absence of any scanner based illumi-

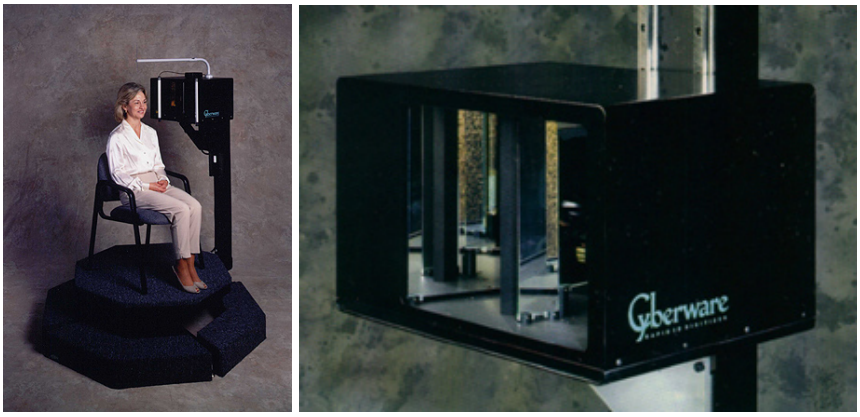

Figure 1.4: Structured light scanners

nation. The most obvious difference is that structured light is mostly applicable to indoor sensing as the projected pattern is usually washed out in sunlight.

For structured light scanners, a projector is used to shine a known imaging pattern onto the scene. This pattern is distinct in some way so that the correspondence is easy to determine. For instance, the Cyberware 3030 PS scanning system (shown in figure 1.4) uses a projected laser line and camera pair to scan depth for a single 2D slice of the object. The scan-head is moved in a circle around the object to produce a cylindrical scan. The Microsoft Kinect sensor on the other hand, projects a fixed known random dot pattern onto the scene and uses constrained search combined with the knowledge of the dot pattern's makeup to determine the correspondence. Figure 1.5 shows the Kinect dot pattern and imaging hardware.

The depth measurements produced from stereo and structured light have some intricacies. They generally tend to be more uniform when viewed from the scanner location than those scans produced from sweeping 2D line scanners. They also tend to have missing pixels near occlusion borders because the object will frequently occlude one camera from the other causing the halo effect seen in figure 1.5.

These scanners produce a different sort of data from image cameras that are so prevalent

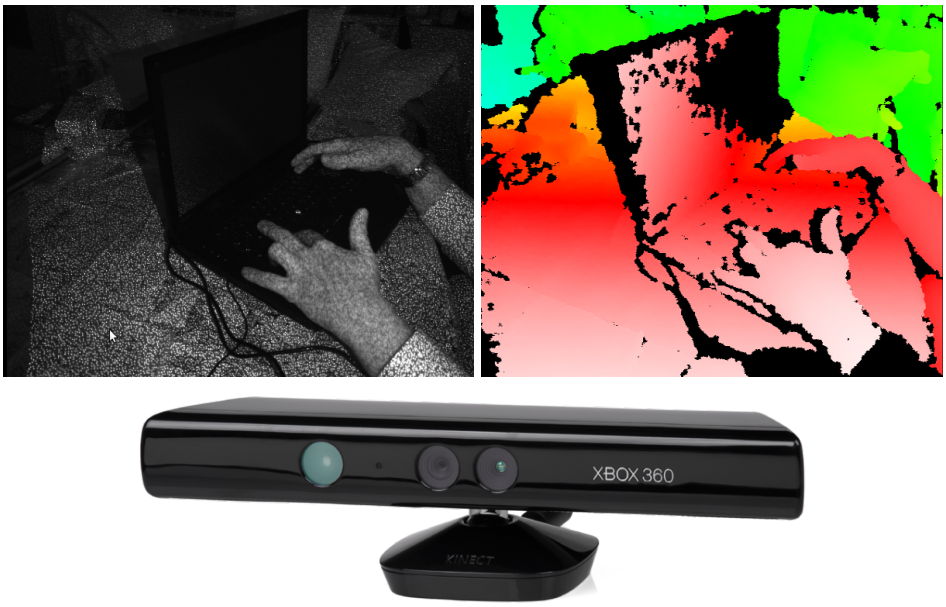

Figure 1.5: The top row shows the Microsoft Kinect structured light pattern and the resulting depthmap, and the bottom image shows the sensor itself. The random dot pattern is projected from the leftmost position while the rightmost is the IR camera which views the dot pattern. The center camera captures an accompanying visible image.

in computer vision. Like all sensors there are advantages and disadvantages, and we hope to show in this thesis that the advantages gained from using depth sensors are significant.

1.2 Iterative Closest Point

Registration is defined as the process of aligning two scans so that they are lined up with one another. The Iterative Closest Point (ICP) algorithm [Besl and McKay, 1992] and its variants [Rusinkiewicz and Levoy, 2001] are considered the gold standard for point cloud registration. We will frequently refer to ICP throughout this thesis, and it is necessary to explain the algorithm. Figure 1.6 illustrated the process of registration.

In order to define registration more concretely we use the partial Procrustes superimpo-

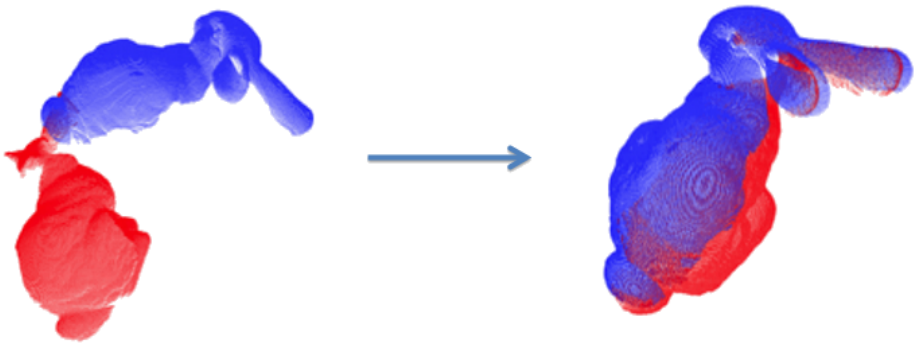

Figure 1.6: Point cloud registration is the process of aligning two scans so that they line up. The left image shows to unaligned scans (red and blue) and the right image shows the aligned version.

sition. We define two point sets X and Y, and write the error

$$error = \min_{R,t} \sum_{(i,j) \in C} ||x_i - (Ry_i + t)||.$$

$x_i \in X$ and $y_j \in Y$, (R, t) defines a rigid transformation, and C is the correspondence set. The correspondence set is a mapping from points in X to points in Y such that each point maps to its nearest neighbor in the truly aligned coordinate system. This C is the main challenge for ICP because generally it is unknown and therefor is part of the solution for the minimization process. ICP chooses an initial alignment from the user interaction or some other method, proceeds to iterate between choosing C based on nearest neighbors in the current alignment, and solves for R, t given C. There are many nearest neighbor libraries [Arya et al., 1998; Muja and Lowe, 2009] that efficiently compute the nearest neighbors using one of a number of methods of which the KD-tree is the best known.

ICP has an few extensions that differ in how they compute the motion (R, r). A recent example is that of Generalized-ICP [Segal et al., 2009] where the local shape is computed via the point density (plane, line, point) and then those ellipsoids are used in the motion estimation process. Another extension is where the motion is computed differently by first

computing normals for one set of points and then minimizing the point to plane distance,

$$error = \min_{R,t} \sum_{(i,j)\in C} n_i(x_i - (Ry_i + t)),$$

to perform the transformation step.

1.3 3D Descriptors

A sub-task of 3D data processing is that of matching regions or sections of scans. The matching process generally happens between two point sets X and Y. These two sets can come from the same scan or different scans, but they are sets of points that we would like to find matching regions in. The matching could also be exact as the case for doing scan registration, or it could be a fuzzy matching if we are looking to recognize objects.

A descriptor is a method for computing some statistics of a region of a scan so that the region may be more easily compared to other regions. A naive descriptor would be to use the set of points themselves and then compare them using ICP described in section 1.2. However, this tends to be a long process and prone to errors since ICP requires accurate initialization and requires mostly overlapping scans. Instead descriptors can be used where a set of statistics are computed about each scan before the comparison step and then those statistics are then compared directly. This is ideally a simple comparison although sometimes it is more involved. Ideally the statistics will lie in a vector space and can be compared using any number of machine learning techniques.

The other advantage of using descriptors is that the statistics may actually describe the shape better in some desirable metric. For instance, if a robot were to look for tennis balls, then precomputing statistics related to roundness of points or curvature would transform the search for tennis balls into a search for a region with uniform nonzero curvature.

We now break descriptors into two distinct categories. In one group are so called global descriptors which are computed on some large scale selection of points. They might be computed on the output of a segmentation algorithm or maybe an entire range scan. Global descriptors are generally not centered around a specific point but instead describe an entire object. Local descriptors on the other hand are centered on a specific point, and they contain information from a neighborhood which is usually determined by selecting points within a radius of the center point. Local descriptors usually contain background information as well which will sometimes help in identifying points of interest. For instance, in order to recognize cars, which are usually located on roads near curbs, descriptors containing the ground are helpful in detecting cars.

We will describe some available descriptors; however, it will help to keep in mind the following general aspects of feature descriptors:

- **Descriptor invariance** - If the object is rotated or moved in some way relative to the scanner or pertinent reference frame, does the descriptor also change?

- **Comparability** - How easy is it to compare two descriptors or a number of descriptors? Ideally the descriptors will lie in a high dimensional space and we can compare them using tools, but sometimes we only have a method for computing distance between descriptors in which case comparison becomes much harder.

- **Discriminability** - How much does the descriptor help with the task at hand? If we compute the descriptor for a bunch of examples inside and outside of our chosen class does it produce examples that are easily separated?

We will describe some of the descriptors that are available in the Point Cloud Library software package [Aldoma et al., 2012].

1.3.1 Local Descriptors

Generally local descriptors will have a feature point about which the descriptor is calculated, and a region of influence defining a neighborhood. The radius may be defined in terms of units of length or it may be defined as a number of neighbors. The choice is a trade-off since a fixed number of neighbors means that histograms will have the same total magnitude but will also be more susceptible to differences in point density.

Another aspect of local features is how they are invariant or variant to rigid transformations. Some feature descriptors are invariant to object transformations. For instance, if an object is rotated and transformed in space the descriptors will maintain their value and continue to match. This is not the same as being able to compute a fully defined transformation between the descriptors however. For example, Spin Images produce a transformation invariant description, but there is still an ambiguity about the normal when aligning the points. Other descriptors such as the 3D Shape Context are not invariant to rotations and must be computed at a number of fixed rotations in order to match an object with an unknown rotation.

There are methods for aligning descriptors. The first is to use the normal and possibly principle curvature of the center point. The normal will give a frame of reference up to a rotation about that normal which will be enough for a rotationally invariant descriptor like Spin Images. But for a rotationally variant descriptor the full transformation must be matched. The options here are to either compute the descriptor at a number of rotations, choose another reference direction such as the world up vector, or use something like the principle curvature which is the direction that the normal is changing the most as one traverses the surface.

Local feature descriptors contain a radius parameter. While this parameter is somewhat robust, the size does need to be tuned for the specific class of object that is being recognized. If the size is too small, the descriptor will describe basic features like planes and corners

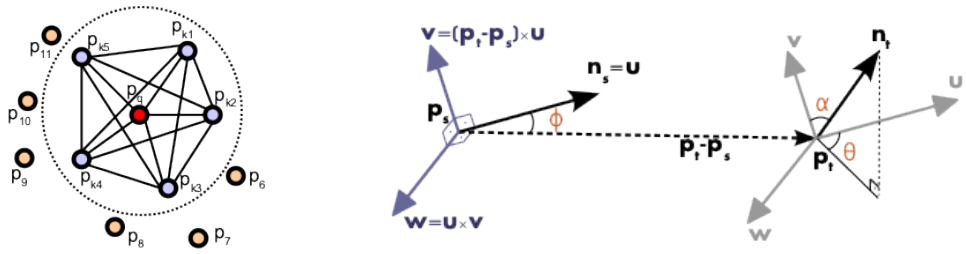

Figure 1.7: The left image shows an illustration of the influence region for a feature point in red and its k-neighbor support region in blue. The fully connected graph is shown. The right image shows the $\mathbf{uvw}$ reference frame for points $\mathbf{p}_s$ and $\mathbf{p}_t$ with normals $\mathbf{n}_s$ and $\mathbf{n}_t$. [Rusu et al., 2009]

and will lose its discriminability. On the other hand, if the radius is so large it will contain information from much of the background and it will not match to anything.

Fast Point Feature Histogram (FPFH)

The FPFH feature descriptor of [Rusu et al., 2009] relies on the center point having a well defined normal, and possibly well defined principle curvature. Principle curvature may not be defined in the case of planes or bowl shapes where the curvature is similar in all directions; however, in many cases there are plenty of features with well defined surfaces and principle curvature directions. Moreover the FPFH only uses principle curvature to register feature keypoints instead of using it in matching the descriptors themselves. Therefore, lack of principle curvature would mean that matching descriptors do not produce a rigid transformation between them.

To define the FPFH descriptor, figure 1.7 shows a center point $\mathbf{p}_q$ and includes the k nearest neighbors as support. The fully connected graph is then computed and for each edge i, j such that $i \neq j$, and the angle between $\mathbf{n}_i$ (the normal at point i) and the line connecting the points is smaller than $\mathbf{n_j}$ and that line. First, we define the "Darboux" $\mathbf{uvw}$

reference frame using the normal and connecting line between the pair of points:

$$\begin{aligned} \mathbf{u} &= \mathbf{n}_i, \\ \mathbf{v} &= \frac{(\mathbf{p}_j - \mathbf{p}_i) \times \mathbf{u}}{||\mathbf{p}_j - \mathbf{p}_i||}, \\ \mathbf{w} &= \mathbf{u} \times \mathbf{v}. \end{aligned} \tag{1.1}$$

Then we define three angles α, ϕ, θ:

$$\begin{aligned} \alpha &= \mathbf{v}^\top \mathbf{n}_j, \\ \phi &= \frac{\mathbf{u}^\top (\mathbf{p}_j - \mathbf{p}_i)}{||\mathbf{p}_j - \mathbf{p}_i||}, \\ \theta &= \arctan(\mathbf{w}^\top \mathbf{n}_j, \mathbf{u}^\top \mathbf{n}_j). \end{aligned} \tag{1.2}$$

Finally, these angles are used to compute a histogram. The bins however are computed by doing independent histograms of each of the angles with 11 bins, and then concatenating the result for a 33 value histogram.

This feature descriptor has the property of being invariant to rigid transformation, and even if the principle curvature direction is not well defined, the descriptor can still be computed and becomes invariant to rotation about the normal of the anchor point. This means that two of these degenerate features can still be matched using descriptors and can be aligned to each other up to an unknown rotation about their normals.

Spin Images

Spin images are discussed in greater detail in section 4.3, but here we will run through their properties. The center point and normal associated with that point ($\mathbf{o}$ $\mathbf{n}$) are used to orient the spin image. Then, for every point within the region of influence $\mathbf{p}_i$ we compute the distance to the normal and the distance to the tangent plane. The distance to the normal following two quantities are calculated using every point within the region of influence. We

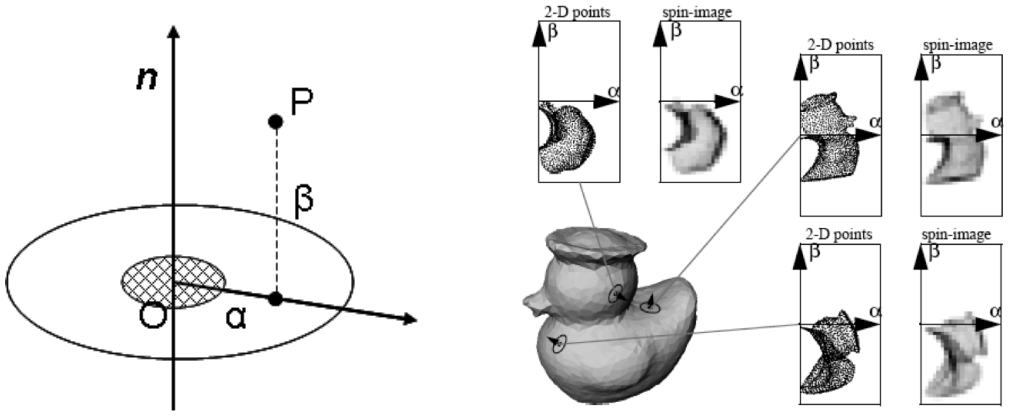

Figure 1.8: The left image shows the quantities α and β and the right image shows examples of spin images computed for a duck mesh.

call these quantities α and β:

$$\begin{aligned} \alpha_i &= ||(\mathbf{p}_i - \mathbf{o}) \times \mathbf{n}||, \\ \beta_i &= \mathbf{n}^\top (\mathbf{p}_i - \mathbf{o}). \end{aligned} \tag{1.3}$$

The algorithm then computes a histogram usually with binning of ten bins in each direction, and in this case the binning is the full 2D histogram of 100 bins. The name "Spin Image" comes from a description of the binning process whereby a sheet of paper is attached to the normal at one end and it is spun around the normal vector splatting all of the support points onto the sheet creating an image.

Signature of Histograms of OrienTations (SHOT)

The SHOT descriptor [Tombari et al., 2010a] attempts to mimic the SIFT descriptor but for the 3D case. First, a local repeatable reference frame is produced using the support points. The method is similar to the Singular Value Decomposition, but it is changed to produce repeatable signs and a more robust reference frame in the presence of noise. Next, the support points are placed into a set of subdivided spherical bins. Finally, for each bin we

take the histogram of orientations. There is also a step of quadrilinear interpolation when placing points into bins because the descriptor tends to change markedly as the points move from bin to bin.

3D Shape Contexts

The 3D Shape Context descriptor [Frome et al., 2004] is similar to the SHOT descriptor above, but it ignores the normals within each bin. It also does not have such a sophisticated method for computing a local reference frame as Tombari et al. [2010a]. Instead it merely uses the normal to align the north pole of the binning scheme, and produces a number of descriptors equal to the number of bins around the equator. There is an extension in which Tombari et al. [2010b] computes a local reference frame using the principle components of the local point distribution to decrease the memory footprint required by replicating the descriptor along the equatorial binning direction.

1.3.2 Global Descriptors

Unlike local descriptors, global descriptors have the support region defined as the input. This may help things if the segmentation is easy, but it also may hurt since segmentation is not generally an easy problem. Regardless, given a selection of points, the task of the descriptor is to compute a high dimensional representation of the set of points. Ideally this representation is robust to occlusions (missing points) and changes in density from scanning patterns and point of view. One aspect which is frequently glossed over in most descriptor papers is any mention of whether or not point densities are computed or even used. For the most part, performance is improved by including the density of points which serves to help match patches that were scanned from near vs. far sensors. Some experiments may scan objects in a turntable or from the same distance, but our experiments with city scale data indicate that differences in density can be significant.

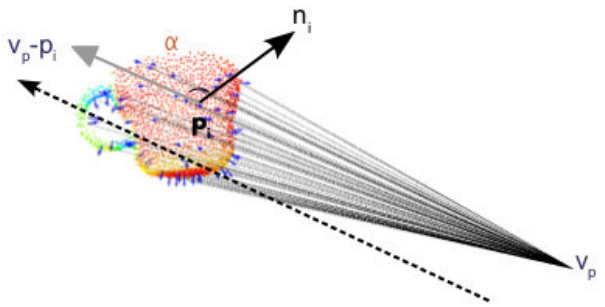

Figure 1.9: The VFHs viewpoint component consists of computing the angle α between the vector connecting the camera and the object centroid $(\mathbf{v}_p - \mathbf{p}_i)$ and the normal for the given support point $\mathbf{n}_i$. Note that the centroid is used for computing the the view vector but the individual normal is used for the support point.

The object support region will generally be determined by a segmentation algorithm or by using an entire scan depending on the problem. A simple segmentation would be the tabletop segmentation algorithm from the Point Cloud Library (PCL).

Point Feature Histogram (PFH)

The PFH [Rusu et al., 2008] is global analog to the FPFH described in section 1.3.1. The same quantities α, ϕ, θ are computed over the fully connected graph on the entire segment or object which does limit the size of data this feature can handle. Instead of concatenating distinct histograms for each of the three dimensions, they are treated as a vector space. The fact that more points are used in the global version allows for the extra number of bins to be useful. The authors did some extra analysis of a distance value as well and determined that the distance did not help discriminability so they removed it from the descriptor. The PCL implementation of PFH leaves out the distance.

Viewpoint Feature Histogram (VFH)

The VFH expands upon the FPFH and PFH but with the goal of determining object class as well as the full 6 degree-of-freedom pose. Rusu et al. [2010] computes a centroid and

average normal for the collection of points, computes the FPFH using the centroid and average normal as the anchor point, and then adds a viewpoint component. The viewpoint component is the angle between the centroid of the object in the sensors coordinate frame and the normal of the support point being added. This geometry is illustrated in figure 1.9. It is then histogrammed more finely than the other angles. The vector between the point associated with the normal can't be used because it will change the feature descriptor depending on the distance to the object. Finally, the descriptor computed from the viewpoint component and the descriptor computed from the spatial component are concatenated into a large histogram containing 250-300 which are then normalized.

VFH shows promise in detecting objects along with pose. But it has a few shortcomings. The use of the viewing angle causes the descriptor to be invariant to rotations about the viewing angle which could be problematic if the goal is to pick up an object. Also, the normalization step causes scale invariance which is generally thought of as bad in 3D data where scale is always available. This is in contrast to camera images where scale invariance is usually thought of positively since the same object farther from the camera will have a different apparent scale. Also the VFH is particularly sensitive to occlusion as missing points will change some number of bins in the descriptor which will then be normalized without them.

As with many descriptors there is a trade-off between discriminability and training size. A descriptor that has strong discriminability will need much more training that one with a lower discriminability. In the case of the VFH, one needs to train using examples from all poses of the objects in question that are to be recognized. These extra poses must be placed into a large classifier which will take up a lot of memory. On the other hand, smaller sized features with more generalizability may make more classification mistakes.

Figure 1.10: Two examples of clustered smooth regions for computing the CVFH.

Clustered Viewpoint Feature Histogram (CVFH)

The last feature we describe had the best performance in the work of Aldoma et al. [2012]. The CVFH expands on the VFH with another segmentation step that splits the object into K locally smooth regions and then computes K VFH's. Figure 1.10 shows some of the segmentations of this algorithm. The missing points problem is addressed by breaking the VFH's apart so that occlusions will only affect some of the clusters. Additionally, the feature is again augmented with a measure of the distance between points normalized by the maximum distance to all support points. This distance does not require setting any global scale and produces an estimate of the elongatedness of the segment, which is helpful since separating out locally smooth regions will decrease discriminability of descriptors.

1.4 Background

We start with the background for 3D Registration and Object Detection. There has been significant work both before and after our contributions to this field. We break up the work between registration and object recognition, but much of the work in the field blurs the line

between the two topics.

1.4.1 Registration Methods in 3D

Many approaches dealing with scans with low overlap forgo global characteristics in favor of the extraction of local invariant features [Huber and Hebert, 2003; Johnson and Hebert, 1997; Stamos and Leordeanu, 2003]. These features, if given in sufficient number, can be matched to constrain the motion estimates. These feature matching approaches are susceptible to outliers and common ambiguities in the matching (repeated textures).

The representation we explore in Chaper 3 is the Extended Gaussian Image (EGI), which can effectively be approximated by a spherical histogram of surface orientations. Since its introduction, a number of other translation invariant spherical representations have been introduced, including extensions to the EGI to handle a wider range of input scans. There are the weighted principal directions and canonical length used in Adan et al. [2001], the directional histogram model [Liu et al., 2003] (and closely related thickness histogram [Liu et al., 2004]), and the spherical attribute images [Delingette et al., 1993; Hebert et al., 1995]. In Kang and Ikeuchi [1993] a complex EGI was proposed which extends the traditional EGI to distinguish between convex and nonconvex objects. Although invariant spherical representations have been used to estimate relative orientation ([Brou, Winter 1984; Hebert et al., 1995; Ikeuchi, 1983; Little, 1985]), these methods depend on unreliable local features or brute force matching.

While our use of spherical harmonics to estimate rotation from EGIs is new, harmonic invariants have been used extensively for object retrieval and recognition [Kazhdan et al., 2003; Liu et al., 2003, 2004], and also at a smaller scale to generate invariant keypoints [Frome et al., 2004]. A true Fourier-based method for range alignment is given in Lucchese et al. [2002]. Since this method estimates the parameters of motion directly from the frequency domain, it requires knowledge of the overlapping regions between scans.

Close methods to ours may be found in the SLAM literature, where correlation alignment is achieved by recovering the phase shift from two dimensional signals. For example, angle histograms, which are roughly invariant to rotation and translation are aligned via cross-correlation in Weiß et al. [1994].

In addition to aligning limited overlap point clouds, another objective of ours is to seamlessly integrate a large number of scans. Related to this effort are a number of methods which try to create object models from the combination of numerous laser scans [Beraldin et al., 1997; Curless and Levoy, 1996; Saucy and Laurendau, 1995]. Curless and Levoy [1996] combines range scans through an updated signed distance function, and in Saucy and Laurendau [1995] the surfaces are integrated by minimizing the least-squares distance between overlapping regions.

Since our work has been published [Makadia et al., 2006], the field has continued to mature. Arguably the most talked about work was that of Newcombe et al. [2011]. They perform real time object mapping and model building using the Kinect Sensor, and while their work includes a method for fusing range images, it also must estimate sensor motion. They do this using a signed distance function and assuming that motions will be small between successive frames (their algorithm runs at 30Hz on the GPU). They solve the same problem of Chapter 2 but for Iterative Closest Point (ICP) in which motion is estimated between each point and the zero crossing of the signed distance function. They use the Euler Angle mapping to solve it which is acceptable because all motions are assumed to be small. The work of Osteen et al. [2012] uses the EGI matching scheme as a pre-processing step along with ICP in order to estimate robot ego-motion, and is robust to large motions between frames allowing for more computationally expensive algorithms or slower computers (It does not require an onboard GPU).

There has also been work which uses locally computed features along with RANSAC or Generalized Hough Transform techniques in order to compute registrations. The work

of Glover et al. [2011] computes local features on candidate objects and then uses the matching feature center points along with principle curvature direction in order to produce alignments which are then clustered using a Generalized Hough Transform. Papazov and Burschka [2011] uses semi-local features in which pairs of normals are combined into a feature, but the pair can be distant. Then these pairs are used to hypothesize correspondences which are verified similarly to our verification step in Section 3.5.

There is also work in performing registrations as optimizations over the space of correspondences. The work of Maciel and Costeira [2003] performs registration, but allows for general affine transformations when solving for 3D-3D registrations.

There are other methods for registration using local features. A good survey is presented in Aldoma et al. [2012] which includes most currently available local features that are used in registration including the Fast Point Feature Histogram (FPFH), Signature of Histograms of Orientations (SHOT), 3D Shape Context, Unique Shape Context, Spin Images, and the Radius-Based Surface Descriptor. Another local feature used for registration is Scale-Dependent/Invariant Features [Novatnack and Nishino, 2008].

1.4.2 3D Object Detection

In this section, we briefly overview related work on local and global 3D shape descriptors and 3D object recognition, focusing only on shape-based descriptors. Research on appearance-based recognition has arguably been more active recently, but is not directly applicable in our experimental setup.

Global shape descriptors include EGIs [Horn, 1984], superquadrics [Solina and Bajcsy, 1990], complex EGIs [Kang and Ikeuchi, 1993], spherical attribute images [Hebert et al., 1995], and the COSMOS [Dorai and Jain, 1997]. Global descriptors are more discriminative since they encapsulate all available information. On the other hand, they are applicable to single segmented objects and they are sensitive to clutter and occlusion. A global repre-

sentation in which occlusion is explicitly handled is the spherical attribute image proposed by Hebert et al. [1995].

A method to obtain invariance to rigid transformations was presented by Osada et al. [2002] who computed shape signatures for 3D objects in the form of shape statistics, such as the distance between randomly sampled pairs of points. Liu et al. [2003] introduced the directional histogram model as a shape descriptor and achieved orientation invariance by computing the spherical harmonic transform. Kazhdan et al. [2003] proposed a method to make several types of shape descriptors rotationally invariant, via the use of spherical harmonics.

Descriptors with local support are more effective than global descriptors for partial data corrupted by clutter. Stein and Medioni [1992] combined surface and contour descriptors, in the form of surface splashes and super-segments, respectively. Spin images were introduced by Johnson and Hebert [1999] and are among the most popular such descriptors. See Section 4.3 for more details. Ashbrook et al. [1998] took a similar approach based on the pairwise relationships between triangles of the input mesh. Frome et al. [2004] extended the concept of shape contexts to 3D. Their experiments show that 3D shape contexts are more robust to occlusion and surface deformation than spin images, but incur significantly higher computational cost. Huber et al. [2004] propose a technique to divide range scans of vehicles into parts and perform recognition under large occlusions using spin images as local shape signatures.

Local shape descriptors have been used for larger scale object recognition. Johnson et al. [1998] use PCA-compressed spin images and nearest neighbor search to find the most similar spin images to the query. Alignment hypotheses are estimated using these correspondences and a variant of the ICP algorithm from Besl and McKay [1992] is used for verification. Shan et al. [2006] proposed the shapeme histogram projection algorithm which can match partial objects by projecting the descriptor of the query onto the sub-

space of the model database. Matei et al. [2006] find potential matches for spin images using locality sensitive hashing. Geometric constraints are then used to verify the match. Ruiz Correa et al. [2006] addressed deformable shape recognition via a two-stage approach that computes numeric signatures (spin images) to label components of the data and then computes symbolic signatures on the labels. This scheme is very effective, but requires extensive manual labeling of the training data. Funkhouser and Shilane [2006] presented a shape matching system that uses multi-scale, local descriptors and a priority queue that generates the most likely hypotheses first.

In most of the above methods, processing is mostly bottom-up, followed in some cases by a geometric verification step. A top-down approach was proposed by Mian et al. [2006] who represent objects by 3D occupancy grids which can be matched using a 4D hash table. The algorithm removes recognized objects from the scene and attempts to recognize the remaining data until no additional library object can be found.

Our method can detect cars in real scenes in the presence of clutter and sensor noise. Very few of the papers mentioned above ([Carmichael et al., 1999; Johnson et al., 1998; Matei et al., 2006; Mian et al., 2006; Ruiz Correa et al., 2006]) present results on real data. Among the ones that do, Matei et al. [2006] classified cars that had been previously segmented. Johnson et al. [1998], Carmichael et al. [1999] and Mian et al. [2006] show object detection from real scenes containing multiple objects. It should be noted, however, that the number of objects in the scene is small and that all objects were presented to the algorithm during training. Ruiz Correa et al. [2006] are able to handle intra-class variation, at the cost of large manual labeling effort. The goal of our work is more ambitious than [Carmichael et al., 1999; Frome et al., 2004; Johnson et al., 1998; Matei et al., 2006; Mian et al., 2006; Shan et al., 2006] in order to make more practical applications possible. Our algorithm is not trained on exemplars identical to the queries, but on other instances from the same class. This enables us to deploy the system on very large-scale datasets with mod-

erate training efforts, since we only have to label a few instances from the object categories we are interested in.

Since publication, the field has continued to mature. One example is an implementation of the Implicit Shape Model (ISM) for LiDAR [Velizhev et al., 2012]. ISM uses local features to vote for object detection's in a bottom up way just as our work does, but is more complex in that it uses a codebook to classify specific part types which can cast more informative votes than simple car/not-car features.

There is some work from Halma et al. [2010] and Yang et al. [2011] which detects cars in aerial LiDAR. Halma et al. [2010] uses the gravity vector to orient single spin images, and then uses moments to initialize ICP around car detections.

Somewhat related is the work of Moosmann and Sauerland [2011] which attempts to cluster object types in urban scenes. They also use the fact that many objects are easily segmented out from their support surfaces.

Behley et al. [2012] compare a number of histogram based local object descriptors including Spin Images used by us, and find that histograms with 3D support regions perform best.

There is a work that quickly classifies each LiDAR point into categories such as ground, vegetation, wall, etc.; some examples are Xiong et al. [2011], Behley et al. [2010]. Also Stamos et al. [2012], relies on specific heuristics in order to recognize certain types of objects such as curbs and cars, and allows for fast online processing as the data is acquired from a scanner such as a Velodyne.

There are some voxel based approaches such as Aijazi et al. [2013] who uses super-voxels and local features made up of statistics about each super-voxel including normal directions or RGB statistics in order to classify urban range data into five categories: buildings, cars, roads, poles, and trees.

1.5 Contributions

For this thesis we have produced a series of systems and algorithms which allow for object detection in large scale scans. Many of the algorithms will be useful for other tasks. The specific contributions are as follows:

1.5.1 3D-Camera Calibration

In Naroditsky et al. [2011] we present a method for calibrating a camera with a 2D range scanner. Our contributions are as follows.

- We have found the minimal solution for aligning 3D-point to 3D-plane using six correspondences.
- We demonstrate how to solve the minimal problem for the overdetermined system.
- We have produced a fully automatic calibration system for unattended use.
- We have also produced a user annotated system where accuracy is absolutely critical.

We also demonstrate that our solutions are correct for synthetic data and multiple real world calibrations.

1.5.2 3D Range Alignment

In our work [Makadia et al., 2006] we address the problem of automatically aligning LiDAR scans. We produced a robust system suitable for automatic alignment. The contributions are as follows.

- We introduce the novel Constellation Extended Gaussian Image descriptor.
- We show how to align Constellation EGIs using both the rotation fourier transform and hough voting.

- We demonstrate the ability to decouple estimation of rotation and translation for significantly reduced CEGI comparison complexity.
- We show that alignment between scans with minimal overlap is achievable.

We later demonstrate that the alignment and verification error metric are useful for object recognition in large scale datasets.

1.5.3 Bottom Up Top Down Object Recognition

We show in Patterson et al. [2008] a method that recognizes cars in a citywide terrestrial LiDAR dataset. Our contributions to the field are:

- the combination of bottom-up and top-down processing to detect potential targets efficiently and verify them accurately,
- the capability of performing training on instances that come from the same object category as the queries, but are not necessarily identical to the queries,
- minimal user efforts during training,
- object detection for large-scale datasets captured in uncontrolled environments,
- and accurate segmentation of target objects from the background.

This all runs accurately and runs in less than a day, which is faster than real time when compared to the weeks of scanning.

1.6 Outline

This thesis is organized as follows:

- **Chapter 2:** We begin by describing a camera to LiDAR calibration scheme which uses point to plane correspondences. In particular, we solve the minimal problem involving six correspondences, and we solve the overconstrained system. Finally, we demonstrate stability of the minimal solution on synthetic calibration data, and we demonstrate a calibration tool with real world examples.

- **Chapter 3:** We describe our method for aligning pointclouds using the Constellation Extended Gaussian Image (CEGI). We start with background on the Extended Gaussian Image. Then the creation and matching of the CEGI is described. Finally, we describe our verification scheme and demonstrate on numerous real world scan alignments.

- **Chapter 4:** Finally, we show our method for detecting objects in large scale scans. The bottom up descriptor (Spin Images) are reviewed, then we show that the CEGI can be used to recognize objects using the overlap percentage. We demonstrate robust performance on a very large dataset.

Chapter 2

Automatic Alignment of a Camera with a Line Scan LiDAR System

2.1 Introduction

The problem of calibrating a LiDAR-camera sensor rig is important in robotics applications. A camera provides a dense, color image of the environment, and LiDAR gives a sparse, but accurate, collection of line scans. Fusion of visual and distance information is challenging when there is parallax between the two sensors and when distance consists of a single line scan. Unlike in structured light techniques, the laser is not visible in the image, hence we have to invent a way to associate features on the line scan with features in the image if we want to eliminate a manual selection of the features.

This chapter outlines both a complete solution involving RANSAC, and a more interactive, robust, and simple solution for surveying. The minimal solution for use with RANSAC [Fischler and Bolles, 1981] is presented in detail in Naroditsky et al. [2011]. The algorithm only assumes that the LiDAR and camera contain overlapping fields of view. However, for the purposes of this thesis, we only use the stability tests from Naroditsky et al. [2011]

and otherwise perform the linearized solution which is used for the final results there as well. Synthetic results are presented to show stability of the solutions, and a real calibration is performed and evaluated. A system for inputting correspondences was build and error results are show for a dataset captured in Cyprus.

2.2 Related Work

The closest and most cited work to ours is by Zhang and Pless [2004] who matches a scanline to a checkerboard. When moving a checkerboard, traditional camera calibration [Bouguet, 2006] can extract the normal to the checkerboard with respect to a global camera reference, while detection of a line in the laser profile enables association of 3D-points with the calibration plane. The algorithm starts with a linear initialization, suffering under the well known effects of linearization like finding 3x3 matrices satisfying the data equation and then finding the closest special orthogonal matrix. Mei and Rives [2006] have applied the same principle to catadioptric images but they exploit the association of a 3D-line (in terms of direction and an offset) to a calibration plane. It is worth noticing that the equation associating the plane normal to the 3D-line direction is of the form $\mathbf{n}^\top R\mathbf{d} = 0$, is algebraically the same as ours after eliminating the translation. However, the authors use the association of points similarl to Zhang and Pless [2004].

When a laser system produces a full depth map at once, the only challenge is associating features. In Unnikrishnan and Hebert [2005], which is similar to Zhang and Pless [2004], the association of 3D points to planes extracted from images are used. In Nunez et al. [2009] an IMU enables the registration of line scans into a 3D LiDAR and the relative transformation is found via hand-eye calibration [Horaud and Dornaika, 1995]. Scaramuzza et al. [2007] uses the association of hand-clicked points in a full 3D map with points in catadioptric images.

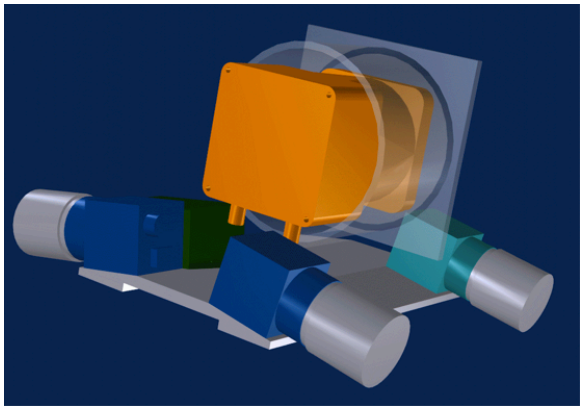

Figure 2.1: A capture rig incorporating four cameras and a Hokuyo LiDAR. Our algorithm automatically calibrates such systems.

2.3 Problem Description

Let us formally define the problem of camera-to-LiDAR calibration. We are given a sensor structure consisting of a calibrated camera (intrinsic parameters are known) and scan line LiDAR that are rigidly mounted with respect to each other. Our goal is to find the rigid transformation $[R|\mathbf{t}]$, where R is a rotation matrix and $\mathbf{t}$ is a translation vector, such that given a 3D point $\mathbf{x} = [x_1, x_2, x_3]^\top$ obtained by the LiDAR, we can compute the corresponding point $\mathbf{y} = [y_1, y_2, y_3]^\top$ in the camera's coordinate system (and then in the image via the intrinsic calibration) as

$$\mathbf{y} = R\mathbf{x} + \mathbf{t}.$$

A single LiDAR datum consists of a depth and angle at which the depth was sensed. We define the coordinate system for the LiDAR as follows. The origin is the center of laser sensor rotation, and the plane of laser rotation is the Y-Z plane. Consequently, we can always express a 3D LiDAR point as $\mathbf{x} = [0, x_2, x_3]^\top$.

As with any calibration problem from sensor data, we must collect correspondences between the readings of different sensors. In this case, we construct a calibration target containing a single black to white transition (see Figure 2.2), which we detect as a line

Figure 2.2: A single camera frame from the calibration data set showing the calibration object. The object consists of a black line on a white sheet of paper. We detect the white-to-black transition looking from the top of the image.

segment in the image and a point in the LiDAR's luminance output for a single line scan (see Figure 2.3). We discuss feature detection in detail in Section 2.5.2. A line in the image corresponds to a plane in the world containing the line and the center of projection of the camera. We now have a correspondence between a 3D point in the LiDAR coordinate system and a plane in the camera's coordinate system. Thus our constraint is that the LiDAR point, taken into the camera's coordinate system, must lie on the corresponding plane. We express this constraint as

$$\mathbf{n}_i^\top (R\mathbf{x}_i + \mathbf{t}) = 0, ||\mathbf{n}_i|| = 1, \tag{2.1}$$

where $\mathbf{n}_i$ is the normal to the plane in the camera coordinate system and $\mathbf{x}_i$ is the LiDAR point for correspondence i. The most accurate calibration is attained when there are many data points, so the solution is computed as an overdetermined set of equations in Section 2.4.

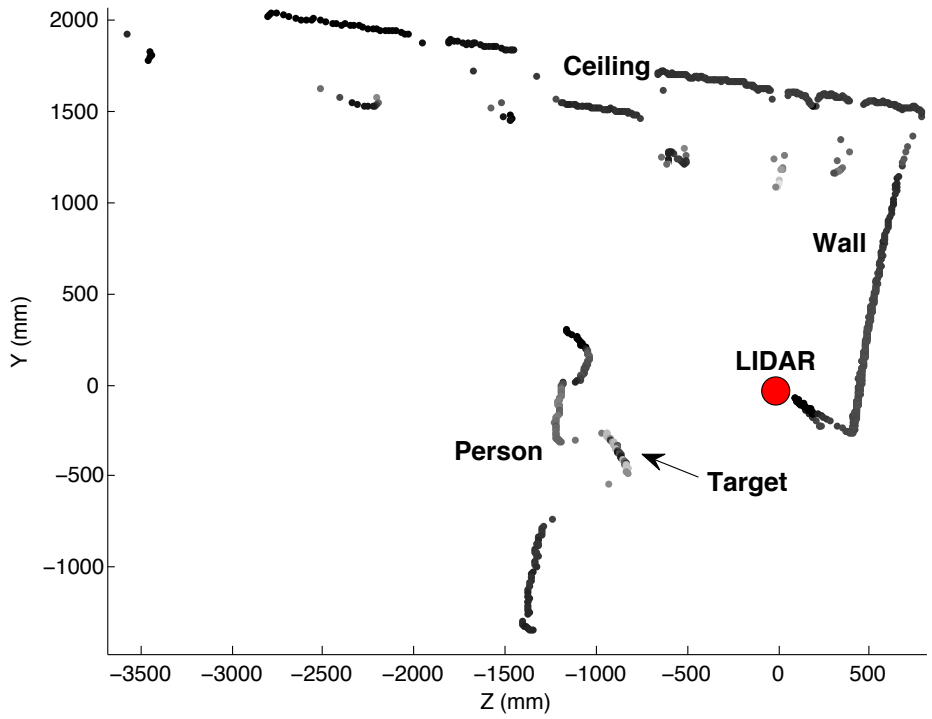

Figure 2.3: A portion of a LiDAR scan showing a person holding the calibration target. The points are colored by their intensity returns. The LiDAR's scan plane is close to vertical, and its origin is marked by a circle.

2.4 Optimization over $\mathbf{SO}(3)$

In order to solve the problem of determining the best rigid transformation for aligning points with planes in 3D we turn to a method which uses the Lie Algebra. Taylor and Kriegman [1994] show how to do this when the optimization variable lies in $\mathbf{SO}(3)$. We will follow this method, but it must be adapted to our specific problem. This is the general outline of our solution:

1. Remove translation $\mathbf{t}$ analytically from the constraints,

2. Solve for R using the Gauss-Newton algorithm similar to Taylor and Kriegman [1994],

3. Compute $\mathbf{t}$ from the value of R via back-substitution.

2.4.1 Removing the translation ($\mathbf{t}$) from the constraints

Recall the equation to solve for m correspondences:

$$\mathbf{n}_i^\top (R\mathbf{x}_i + \mathbf{t}) = 0, \quad 1 \leq i \leq m. \tag{2.2}$$

$[R, \mathbf{t}]$ are the variables over which we would like to optimize. Because there may be noise in the system, the equality will likely not be exact, so instead we convert the problem into an L_2-minimization for the objective function $F(R, \mathbf{t})$ defined:

$$\begin{aligned} F(R, \mathbf{t}) &= \sum_i ||\mathbf{n}_i^\top (R\mathbf{x}_i + \mathbf{t})||^2 \\ &= \sum_i (\mathbf{n}_i^\top (R\mathbf{x}_i + \mathbf{t}))^2. \end{aligned} \tag{2.3}$$

The first step is to remove $\mathbf{t}$ from the optimization. Recall that the minimum value of a convex quadratic function is attained at the point where the derivative of that function is zero. We know the function is quadratic because we can rearrange the $\mathbf{t}$'s in the equation into the form $\mathbf{t}^\top(\mathbf{n_i}\mathbf{n_i}^\top)\mathbf{t}$. And we know that it is convex because all the squared terms are inside a positive sum. So we can take the derivative of $F(R, \mathbf{t})$ with resepct to $\mathbf{t}$, set the result to zero, and use the chain rule:

$$\begin{aligned} \frac{\partial F(R, \mathbf{t})}{\partial \mathbf{t}} = \frac{\partial}{\partial \mathbf{t}} \sum_i (\mathbf{n}_i^\top (R\mathbf{x}_i + \mathbf{t}))^2 &= \mathbf{0} \\ \sum_i 2(\mathbf{n}_i^\top (R\mathbf{x}_i + \mathbf{t})) \frac{\partial}{\partial \mathbf{t}} (\mathbf{n}_i^\top (R\mathbf{x}_i + \mathbf{t})) &= \mathbf{0} \\ 2 \sum_i \mathbf{n}_i^\top (R\mathbf{x}_i + \mathbf{t}) \mathbf{n}_i^\top &= \mathbf{0}. \end{aligned}$$

Subtract $2\sum_i(\mathbf{n}_i^\top \mathbf{t})\mathbf{n}_i^\top$ from both sides and simplify:

$$\begin{aligned}
2\sum_i \mathbf{n}_i^\top(R\mathbf{x}_i+\mathbf{t})\mathbf{n}_i^\top - 2\sum_i(\mathbf{n}_i^\top\mathbf{t})\mathbf{n}_i^\top &= -2\sum_i(\mathbf{n}_i^\top\mathbf{t})\mathbf{n}_i^\top \\
2\sum_i(\mathbf{n}_i^\top R\mathbf{x}_i)\mathbf{n}_i^\top &= -2\sum_i(\mathbf{n}_i^\top\mathbf{t})\mathbf{n}_i^\top \\
\sum_i(\mathbf{n}_i^\top R\mathbf{x}_i)\mathbf{n}_i^\top &= -\sum_i(\mathbf{n}_i^\top\mathbf{t})\mathbf{n}_i^\top \\
\sum_i \mathbf{n}_i\mathbf{n}_i^\top R\mathbf{x}_i &= -\sum_i \mathbf{n}_i\mathbf{n}_i^\top\mathbf{t} \\
\sum_i \mathbf{n}_i\mathbf{n}_i^\top R\mathbf{x}_i &= -\left(\sum_i \mathbf{n}_i\mathbf{n}_i^\top\right)\mathbf{t}.
\end{aligned}$$

Thus, $\mathbf{t}$ can be written in terms of correspondence data ($\mathbf{n}_i$ and $\mathbf{x}_i$) and R. $\sum_i \mathbf{n}_i\mathbf{n}_i^\top$ is invertible when the matrix is full rank. Full rank will be achieved when the data is well conditioned; so, for the general case of many correspondences this won't be a problem. The solution for $\mathbf{t}$ is:

$$\mathbf{t} = -\left(\sum_i \mathbf{n}_i\mathbf{n}_i^\top\right)^{-1}\sum_i \mathbf{n}_i\mathbf{n}_i^\top R\mathbf{x}_i.$$

And for simplicity we define N which depends on the data:

$$N = \sum_i \mathbf{n}_i\mathbf{n}_i^\top, \quad \mathbf{t} = -N^{-1}\sum_i \mathbf{n}_i\mathbf{n}_i^\top R\mathbf{x}_i. \tag{2.4}$$

2.4.2 Solving for R using the Gauss-Newton algorithm

First, we will review the Gauss-Newton algorithm. This algorithm is a method that can be used in solving a minimization problem and is a special case of the Newton method for gradient descent. In general Gauss-Newton can be used when the function $S(\beta)$ can be

written as a sum of squared function values:

$$S(\beta) = \sum_{i=1}^{m} r_i^2(\beta). \tag{2.5}$$

The Gauss-Newton algorithm defines the gradient descent update step:

$$\beta^{(s+1)} = \beta^{(s)} - (J_r^\top J_r)^{-1} J_r^\top r(\beta^{(s)}), \tag{2.6}$$

where J_r is the Jacobian matrix of the function r at $\beta^{(s)}$.

We will now show how we convert our problem into appropriate form for the Gauss-Newton algorithm. First, define vec(R) as the stacked version of R such that the columns are stacked in order making vec$(R) \in \mathbb{R}^9$:

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}, \quad \text{vec}(R) = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}.$$

This identity follows from matrix multiplication, the definition of vec(R), and the definition of the Kronecker Product:

$$\mathbf{n}_k R \mathbf{x}_k = (\mathbf{x}_k^\top \otimes \mathbf{n}_k^\top)\text{vec}(R). \tag{2.7}$$

We can now rewrite Equation 2.4 using the identity 2.7:

$$N = \sum_i \mathbf{n}_i \mathbf{n}_i^\top, \quad \mathbf{t} = -N^{-1} \sum_i \mathbf{n}_i (\mathbf{x}_i^\top \otimes \mathbf{n}_i^\top)\text{vec}(R). \tag{2.8}$$

Recall the objective function $F(R, \mathbf{t})$ (2.3). We can now substitute equation 2.8 into

the objective. We rename the objective $F(R)$ since it no longer depends on $\mathbf{t}$:

$$\begin{aligned} F(R,t) &= \sum_i (\mathbf{n}_i^\top (R\mathbf{x}_i + \mathbf{t}))^2, \\ F(R) &= \sum_i \left(\mathbf{n}_i^\top \left[R\mathbf{x}_i - N^{-1} \sum_j \mathbf{n}_j (\mathbf{x}_j^\top \otimes \mathbf{n}_j^\top) \text{vec}(R) \right] \right)^2 \\ &= \sum_i \left(\mathbf{n}_i^\top R\mathbf{x}_i - \mathbf{n}_i^\top N^{-1} \sum_j \mathbf{n}_j (\mathbf{x}_j^\top \otimes \mathbf{n}_j^\top) \text{vec}(R) \right)^2. \end{aligned}$$

Apply the identity 2.7 one more time and pull out $\text{vec}(R)$:

$$F(R) = \sum_i \left(\left[(\mathbf{x}_i^\top \otimes \mathbf{n}_i^\top) - \mathbf{n}_i^\top N^{-1} \sum_j \mathbf{n}_j (\mathbf{x}_j^\top \otimes \mathbf{n}_j^\top) \right] \text{vec}(R) \right)^2. \tag{2.9}$$

Define a data matrix A which is 9xm and contains all of the data dependent terms from Equation 2.9:

$$A = \left[\mathbf{a}_1 \dots \mathbf{a}_m \right], \quad \mathbf{a}_i = (\mathbf{x}_i^\top \otimes \mathbf{n}_i^\top) - \mathbf{n}_i^\top N^{-1} \sum_j \mathbf{n}_j (\mathbf{x}_j^\top \otimes \mathbf{n}_j^\top). \tag{2.10}$$

Now the objective function can be written compactly:

$$\begin{aligned} F(R) &= \sum_i (\mathbf{a}_i \text{vec}(R))^2 \\ &= \left[A^\top \text{vec}(R) \right]^2 \\ &= \left[A^\top \text{vec}(R) \right]^\top \left[A^\top \text{vec}(R) \right] \\ &= \left[\text{vec}(R)^\top A A^\top \text{vec}(R) \right]. \end{aligned} \tag{2.11}$$

For simplicity, we also define the functions $f(R)$ and $f_i(R)$ as

$$f_i(R) = \mathbf{a}_i \text{vec}(R), \quad f(R) = A^\top \text{vec}(R),$$

which allows the objective to be written as

$$F(R) = \sum_i f_i^2(R) = f(R)^\top f(R). \tag{2.12}$$

Here, we apply the Gauss-Newton method to get the update rule for the parameter space R. In our case, the function $r_i(\beta)$ from Equation 2.5 is $f_i(R)$. We can now write the update rule from Equation 2.6:

$$R^{(s+1)} = R^{(s)} - (J_f^\top J_f)^{-1} J_f^\top f(R^{(s)}),$$

where J_f is the Jacobian of $f(R^{(s)})$.

We would normally just compute the answer, but this is hard because the Jacobian expressions are to be computed in the non-euclidean $\mathbf{SO}(3)$ group of rotation matrices. To solve this we follow Taylor and Kriegman [1994] who use the Lie Algebra $\mathfrak{so}(3)$ to compute the updates and then transforms those updates back into $\mathbf{SO}(3)$ using the mapping between the Lie Algebra and Lie Group.

To summarize, we have an expression for the translation in terms of the rotation and data, and an objective function for the rotation solely in terms of the data. At this point we can't use the same method to solve for R as we did for $\mathbf{t}$ because, unlike $\mathbf{t}$, the constraints on R itself are nonlinear. Instead, we compute the instantaneous derivatives of the objective $F(R)$ in a linear tangent space known as the Lie Algebra ($\mathfrak{so}(3)$) and thus we can use a gradient descent optimization, specifically the Gauss-Newton algorithm.

2.4.3 Using Lie Algebra to find R

Now we show how to update the rotation matrix. They proceed by defining a Lie Algebra ($\mathfrak{so}(3)$) which lies in a Euclidian space and is thus easy to compute Jacobians. This Lie

Algebra also has a simple mapping which can be applied to go from the rotation group ($\mathbf{SO}(3)$) to the algebra and back. Thus using the approximation to the function in the $\mathfrak{so}(3)$ we can compute an optimization estimate for the optimal value and then map that value back into $\mathbf{SO}(3)$.

This parametrization can be written down as the current rotation matrix plus the mapping from parameters to rotation matrices, we start with some notation:

$$\omega = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T, \quad \theta = ||\omega||, \quad \tilde{\omega} = \frac{\omega}{||\omega||}, \quad \hat{\omega} = \begin{pmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{pmatrix}. \quad (2.13)$$

Then we write the parametrization centered at a rotation R_k:

$$R_k(\omega) = R_k \exp\{\hat{\omega}\}, \quad \omega \in \mathbb{R}^3, \ \sqrt{\omega^T \omega} < \pi.$$

The $\exp$ is the exponential map and is is defined:

$$R_k(\omega) = R_k \exp\{\hat{\omega}\} = R_k \sum_{n=0}^{\infty} \{\frac{1}{n!} \hat{\omega}^n\},$$

and ω is restricted to the open ball of radius π. This is not a big restriction in our context as we can always decrease our step size into the unit ball and simply perform another step. The proof that this map is surjective is shown in theorem 1.6 from Gallier [2011]. This can be rewritten using the well known Rodrigues Rotation Formula:

$$R_k(\omega) = R_k(I + \sin(\theta)\hat{\omega}) + (1 - \cos(\theta))\hat{\omega}^2). \quad (2.14)$$

We will need to differentiate this with respect to ω. The derivative with respect to the

first parameter ω_x is:

$$\begin{aligned}\frac{\partial}{\partial \omega_x} R_k(\omega)\Big|_{\omega=0} &= \frac{\partial}{\partial \omega_x}(R_k \exp\{\hat{\omega}\})\Big|_{\omega=0} \\ &= \frac{\partial}{\partial \omega_x}(R_k \sum_{n=0}^{\infty}\{\frac{1}{n!}\hat{\omega}^n\})\Big|_{\omega=0} \\ &= R_0\hat{x},\end{aligned}$$

where $\hat{x}$ is the hat operator defined in Equation 2.13 applied to the x-axis unit vector.

We can proceed following the Gauss-Newton method in order to optimize the objective from Equation 2.12:

$$\delta_{k+1} \leftarrow -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T f(R_k), \tag{2.15}$$

where $\mathbf{J}$ is the Jacobian of $f(R_k(\omega))$

$$\begin{aligned}\mathbf{J} &= A\begin{bmatrix}\text{vec}(\frac{\partial}{\partial \omega_x}) & \text{vec}(\frac{\partial}{\partial \omega_y}) & \text{vec}(\frac{\partial}{\partial \omega_z})\end{bmatrix} \\ &= A\begin{bmatrix}\text{vec}(R_k\hat{x}) & \text{vec}(R_k\hat{y}) & \text{vec}(R_k\hat{z})\end{bmatrix}.\end{aligned} \tag{2.16}$$

Using Equations 2.15 and 2.16 we can compute δ directly. It must be projected onto the $SO(3)$ manifold in order to be composed with R_k to compute R_{k+1} which is done using Rodrigues Rotation Formula Equation 2.14.

2.5 Results

In this section we will first establish the correctness of our algorithm and explore its sensitivity to noise using simulated data, and then perform a real calibration of a LiDAR-camera rig mounted on a mobile robot for the purpose of coloring the 3D LiDAR points with pixels from the camera and show the results. This section uses the results of Naroditsky et al. [2011] in order to show that the proposed method for calibrating LiDAR-Camera is stable

in the minimal case, so it is likely stable for the linearized case too.

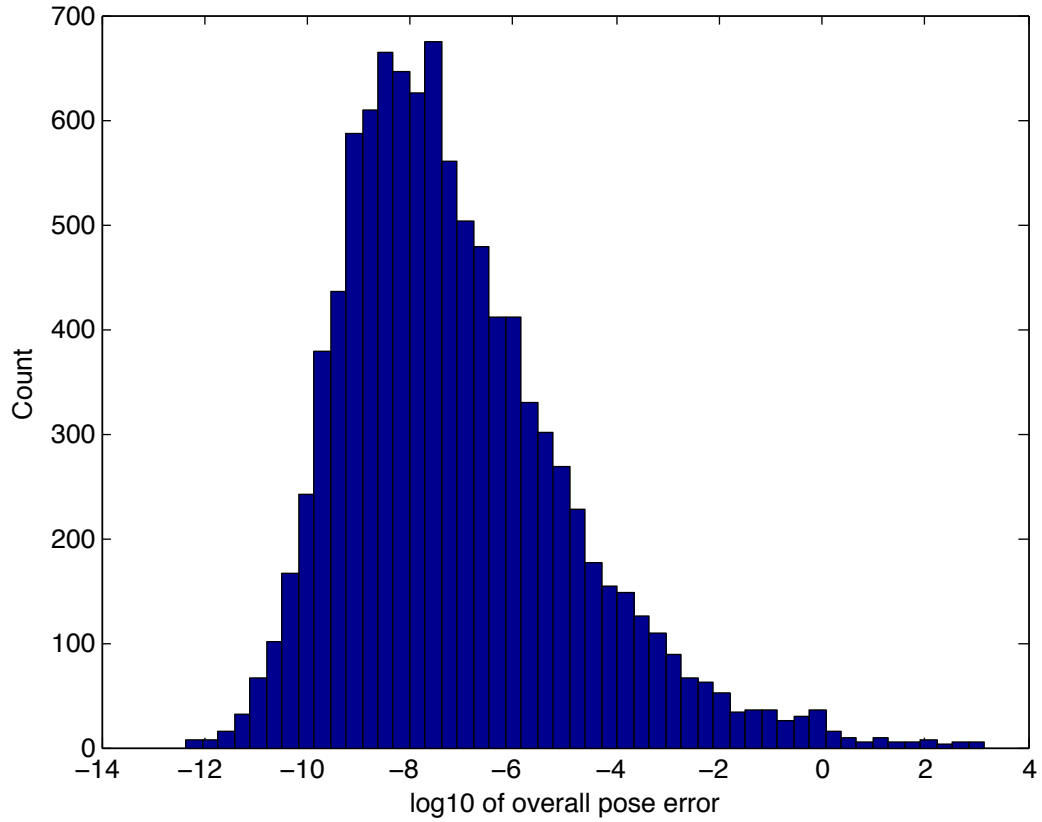

Figure 2.4: The histogram of numerical errors for 10^5 random, noise-free instances of the problem. The error is defined as the $\log_{10} e$, where e of the Frobenius norm of the difference between the ground truth and the computed matrices (see (2.17)). Since the points were not checked for degeneracy (such as collinearity), some failures are observed. If we consider a failure to be $\log_{10} e > -1$ which corresponds to an error of about 0.5° or 1cm, then the method fails 1.97% of the time.

2.5.1 Simulations

Our first experiment establishes the correctness and numerical stability of our symbolic template. We generate, uniformly at random, roll, pitch, and yaw of the LiDAR with respect to the camera in the range of $\pm 30^\circ$, and translation vectors with uniformly random components from 0 to 30cm. For each of these ground truth calibrations, we generate six

noise-free correspondences.

We generate these correspondences by simulating a camera looking at a target. Specifically, we choose two random points in sampling volumes in front of the camera, one on the left and one on the right. This closely models what happens in the real system where the 3D line must intersect the LiDAR scan plane which is close to the vertical plane separating the left and right halves of the image. These two points define a line which is then intersected with the LiDAR plane to obtain the point $\mathbf{x}$. The points, along with the camera center, define the plane and its normal $\mathbf{n}$.

The histogram of errors for 10^5 such configurations is shown in Figure 2.4. The error metric is the following:

$$e = \min_i(\|[R_{gt} \,|\, \mathbf{t}_{gt}] - [R_i \,|\, \mathbf{t}_i]\|_{\mathcal{F}}), \tag{2.17}$$

for i from 1 to the number of solutions, R_{gt} and $\mathbf{t}_{gt}$ comprise the ground truth calibration and $\|\mathbf{.}\|_{\mathcal{F}}$ is the Frobenius norm. The figure demonstrates that our solution correctly solves the calibration problem. The accuracy varies due to the random nature of correspondences and lack of degeneracy checking during sampling.

We choose the Frobenius norm because we need a metric with which to combine both the translational and rotational error into one number. Choice of error metric is a difficult problem to solve since the rotational error and translational error both have different units (meters and degrees). Since in the simulations we are only interested in the case where the errors approach zero we can put both of these units together for the $\|\cdot\|_{\mathcal{F}}$. But this does not work when attempting to quantify a real world calibration, and in that case we estimate rotation and translational error separately.

We now profile the algorithm with respect to noise in sensor data. We consider three sources of error: depth uncertainty in the LiDAR points, misestimation of position, and

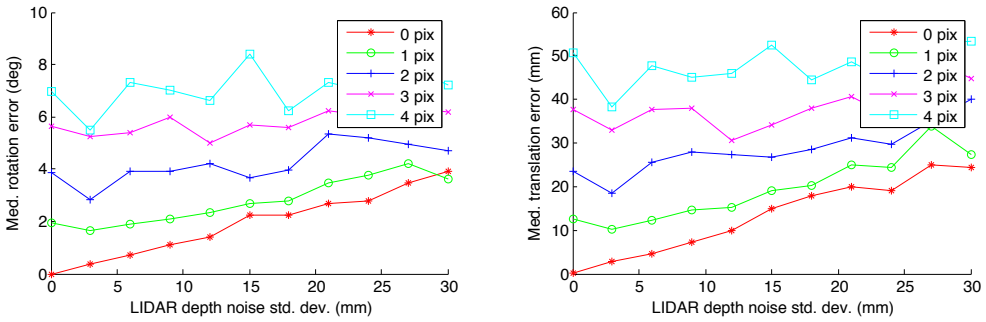

Figure 2.5: Errors in rotation and translation estimation for a simulated rig with 100mm of distance between the camera and LiDAR. Each point shows median error for 200 random configurations of LiDAR-image correspondences. Each sequence corresponds to different levels of image noise plotted against LiDAR noise. The noise values are the standard deviations. The image errors are line translation error (pix) for the baseline camera described in Section 2.5.1.

rotation of the corresponding lines in the image. These all lead to the 3D point being some distance off the plane through the line and center of the camera. For the LiDAR depth error, the Hokuyo UTM-30LX specifies the standard deviation of 30mm for ranges less than 1m. We will study the accuracy for noise standard deviations of 0 to 30mm. The image processing accuracy will be in pixels with respect to a baseline calibrated camera, which we define as a 640×480 pixel sensor with a 60° field of view. While in the real data the errors in line extraction will depend on the length of the edge segments extracted, we will use the range of 0 to 4 pixel standard deviations in the simulated results. The results for different error levels are shown in Figure 2.5, and demonstrate a greater sensitivity to image noise than depth noise.

2.5.2 A Fully Automatic Real Calibration

The sensor rig for the real-world calibration experiment consists of a calibrated 640×480 Flea2 camera with a 77° field of view and a Hokuyo UTM-30LX line scanner, with angular resolution of $0.25\,\mathrm{deg}$ (see Figure 2.1).

Images of the calibration target (see Figures 2.2 and 2.3) are captured synchronously by the two sensors at various positions. We detect the target as follows. For the LiDAR calibration target detection we use the line scan, including intensity, returned from the device in the region of interest for calibration. First, the derivatives of the intensity vector of the line scan are computed using a difference of gaussians filter. The peaks of this derivative signal are detected by non-maximal suppression. This detects both rising and falling edges in the intensity signal. We then improve the estimate of the edges by fitting a line to all the neighboring 3D points and projecting the intensity edge sample point onto that line to give us the final LiDAR feature point $\mathbf{x}_i$. The discrete sampling of the angle could cause an angular error, but since we can control the target's location during calibration, this error can be controlled. Even if the LiDAR and camera are further apart as on a larger robot, a long target can be used which could keep the target close to both sensors.

The first two steps in image line extraction are radial distortion removal and edge detection [Canny, 1986]. The edgels are then combined using the Hough transform to output line segments. We define "linescore" as a measure of how well the gradient information in the image fits with the proposed line. We compute this on the line segments to orient them and prune out ones with poor support. In order to define linescore, first define Q_j as the set of pixels which lie within 1 pixel of the line segment j to score, and define $\mathbf{g}_i$ to be the gradient at pixel i, and $\mathbf{m}_j$ to be the normal to the line segment j which we are scoring. $linescore_j = \sum_{i \in Q_j} \mathbf{g}_i^\top \mathbf{m}_j$ measures the support in the gradient image for each line.

Next, the candidate edges are pruned more using the following heuristics, which use the fact that our target contains a single black stripe on a white background. The heuristics look for pairs of line segments which contain a white to black transition followed by a black to white transition. To find these transitions, we look at the normal direction combined with the linescore to propose candidate pairings respecting this constraint. For each candidate pairing we perform a number of tests:

1. Check that the candidate pair's normals are close to parallel while accounting for possible perspective distortions.

2. The lines are required to be close together because the target never fills up too much of the field of view.

3. Check that the ratio of width to height of the rectangle created by the pair of line segments is appropriate. This is a check that the target has the correct aspect ratio.

4. Overlap is computed and thresholded to rule out line segments which don't occur next to each other.

Once these criteria are passed we take the two endpoints of each line, and record the normal to the plane $\mathbf{n}_i$ passing through these points and the camera center. Thus the vector $\mathbf{n}_i$ corresponding to the LiDAR point $\mathbf{x}_i$ becomes the input for RANSAC process. We then compute a robust calibration solution using around 400 correspondences in the RANSAC framework. The process chooses the best calibration hypothesis based on six correspondences which we then iteratively refine using all of the inliers.

Next we tested the quality of our calibration. Our rig is equipped with stereo cameras that we calibrated using the Camera Calibration Toolbox [Bouguet, 2006]. We computed the error in the LiDAR-camera calibration by observing that we can compute the left-to-right camera calibration by combining the left camera and right camera to LiDAR calibrations. We define an error metric as follows. Let us denote a calibration transformation $P_q = \left[\begin{smallmatrix} R_q & \mathbf{t}_q \\ 0\,0\,0 & 1 \end{smallmatrix}\right]$. We can define the error matrix

$$P_{err} = P_{lr} P_{rh} P_{lh}^{-1},$$

where P_{lr} is the left-to-right calibration computed using the calibration toolbox, and P_{rh} and P_{lh} are the left camera and right camera to Hokuyo calibrations, respectively, computed

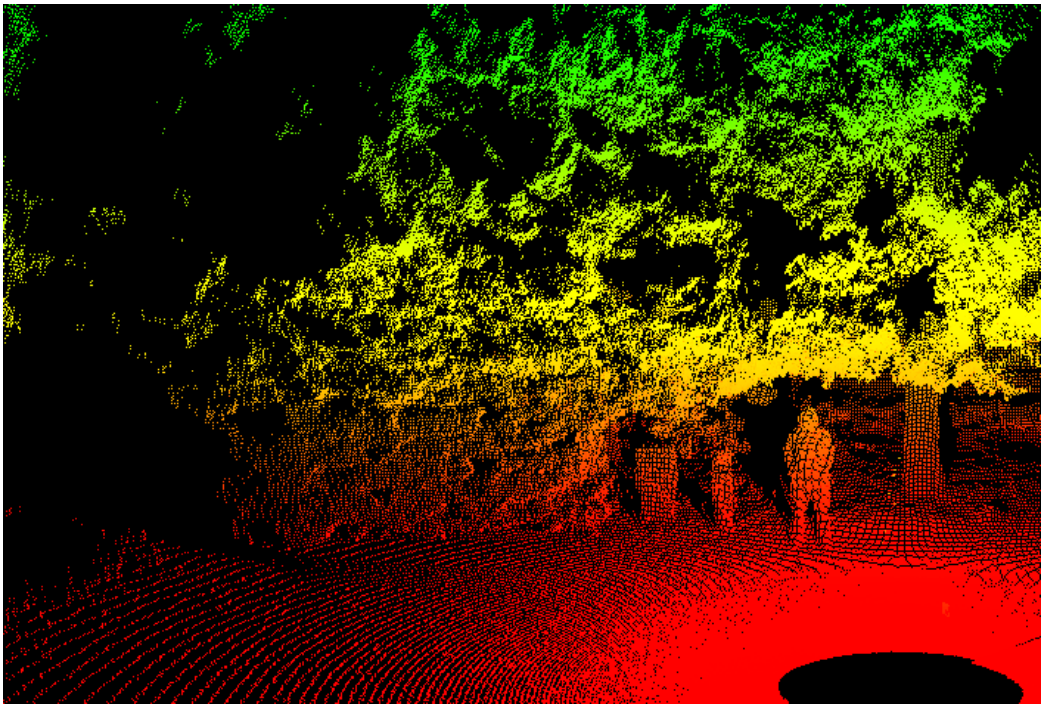

Figure 2.6: A sample LiDAR scan acquired by the mobile robot colored by height. This shows that the cameras visual odometry plus LiDAR-camera calibration is good enough to produce a visually appealling pointcloud.

with our method. If both of our calibrations were accurate, R_{err} would be close to identity and $\|\mathbf{t}_{err}\|_2$ would be close to 0. The real rotation error was $0.26°$, and the translation error was 1.9mm (the distance between the LiDAR and each camera was approximately 140mm).

This calibration was used to color the LiDAR points with the corresponding pixels from the camera. Our system automatically acquired the images and registered LiDAR scans using visual odometry. The registered LiDAR point cloud is shown in Figure 2.6. The same point cloud colored by the camera image pixels is shown in Figure 2.7.

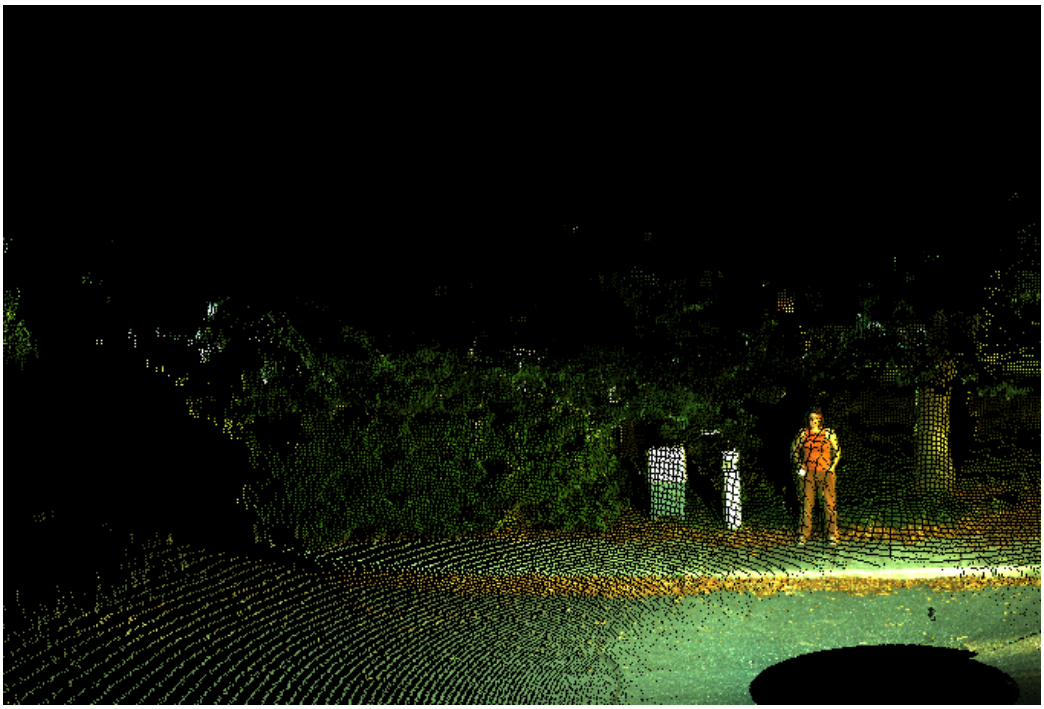

Figure 2.7: A LiDAR scan colored using camera pixels.

2.6 A User Driven Calibration System

We also present a robust application of our system of calibration which was used to accurately scan roughly 500km of the Cypriot highway system. We chose to sidestep issues with RANSAC and thresholding in order to simplify the process of getting a robust calibration. Automatic calibration is frequently toted as a better solution in the literature; however, sometimes the most important deliverable is the integrity of the calibration in which case user interaction is more important than automation. This system was built using six 1.4 megapixel Point Grey Flea2 cameras and two Hokuyo UTM-30LX scanners. It was mounted on top of a car for highway driving. For calibration purposes the cameras were set to capture an image every one second, but because the LiDAR captures images continuously at a frequency of 40 lines per second, any given LiDAR measurement could be out of sync by up to 12.5ms. This will cause errors in measurements, but because enough data

is captured with random movement, the error is assumed to be uniform in all directions. There will be non-uniformity in distance to the camera in terms of pixels, but there is also higher edge detection errors for both the LiDAR and Camera detection algorithms, so these errors are assumed to be small.

In order to calibrate, a user proceeds as follows. First, collect a sequence of a line pattern as described in Section 2.3. Second, process all of the LiDAR data in order to extract all of the black-white transitions as described in Section 2.5. Finally, the user clicks on the images in order to first select LiDAR points and then the detected edges (the same system as Section 2.5 was used). The user clicks on ten or so edges and recomputes the calibration in order to recompute the projection of the LiDAR points in the images. It should be noted that this requires an initial calibration guess which was acquired by hand measuring the LiDAR and camera locations and orientation. The system was quite robust to errors in this calibration of up to 0.5m and 20°.

Figure 2.9 shows the error plots of the final system for one calibration. Because the error is evenly distributed across the image plane and the LiDAR plane it show that, for most LiDAR points detected, they do in fact map to the correctly selected corresponding point and therefore our final calibration is close. Figure 2.10 shows the same error plot, but with respect to pixel error instead of LiDAR error.

2.7 Conclusion

In this chapter we overcome the difficulty of calibrating LiDAR-camera rigs by introducing a new algorithm based on the minimal solution to the calibration from line to 3D point correspondences. We used this algorithm in a robust framework and without initialization. Using the automatic feature detection described, the algorithm can be used to automatically calibrate a variety of sensor platforms. Our experiments with simulated and real data

indicate that this method is both correct and practical. We also presented a method for calibration which uses annotated correspondences that is practical and easy to use because of visual feedback.

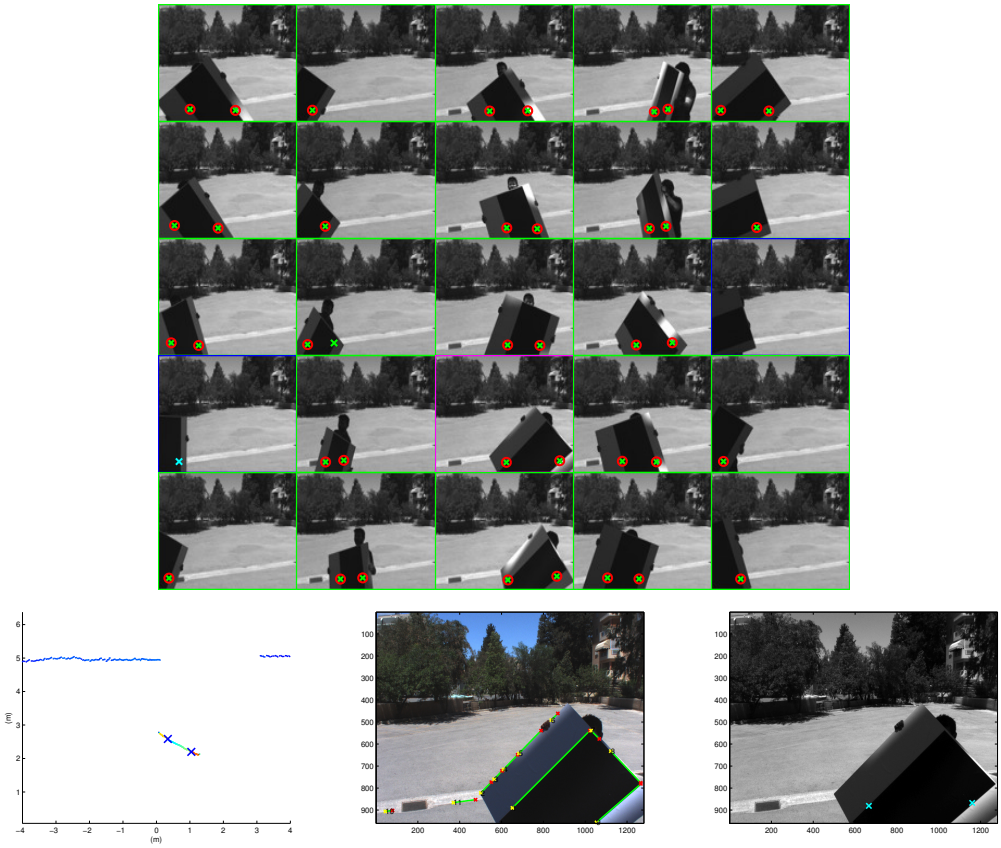

Figure 2.8: An overview of the calibration software system. Counterclockwise from top: a) An overview of 25 frames with x's showing projected LiDAR points in the images. The manually selected points have been circled. b) A top view of the LiDAR showing intensity and automatically detected dark light transitions. c) The camera image with automatically detected lines superimposed upon it. d) The camera image with the LiDAR points projected into it using the cameras intrinsic parameters and the current best estimate of LiDAR-Camera calibration.

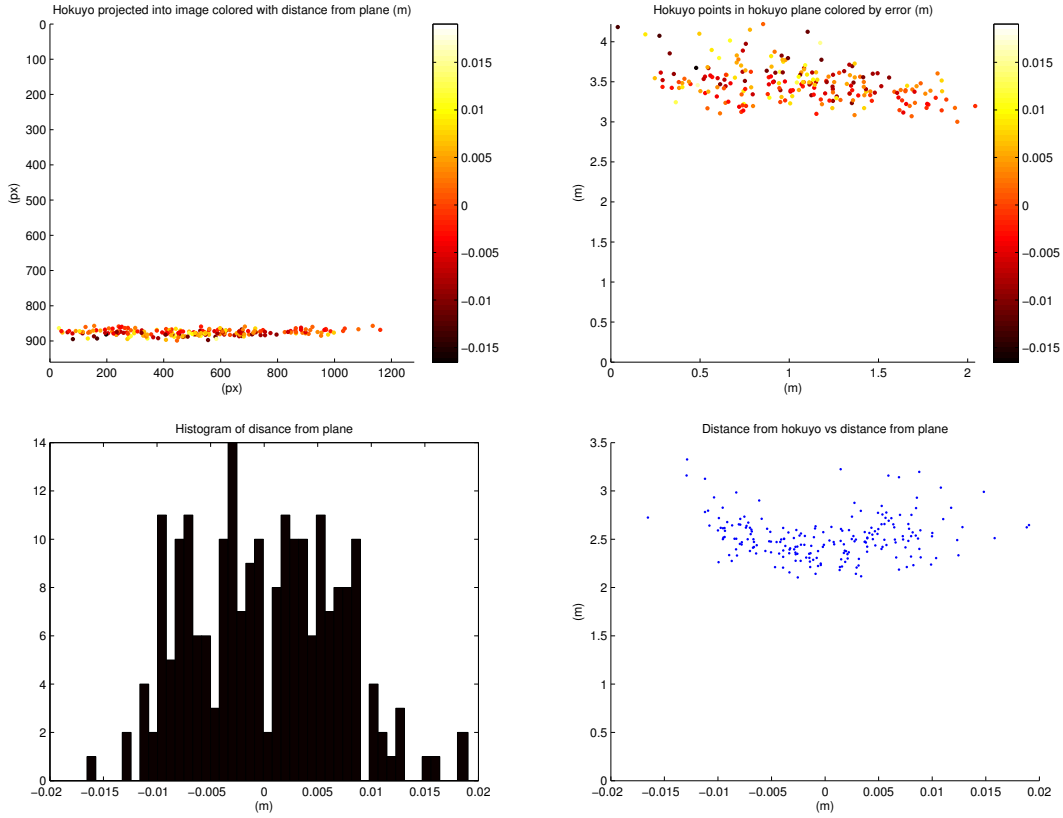

Figure 2.9: Error plots of how this calibration fits the underlying model. Clockwise from top left: a) All of the selected LiDAR points projected into an image plane and colored with their distance from the plane. b) All of the selected LiDAR points in their own plane colored with their distance from the plane. c) The distribution of LiDAR points within their plane. d) A histogram of total error in LiDAR points.

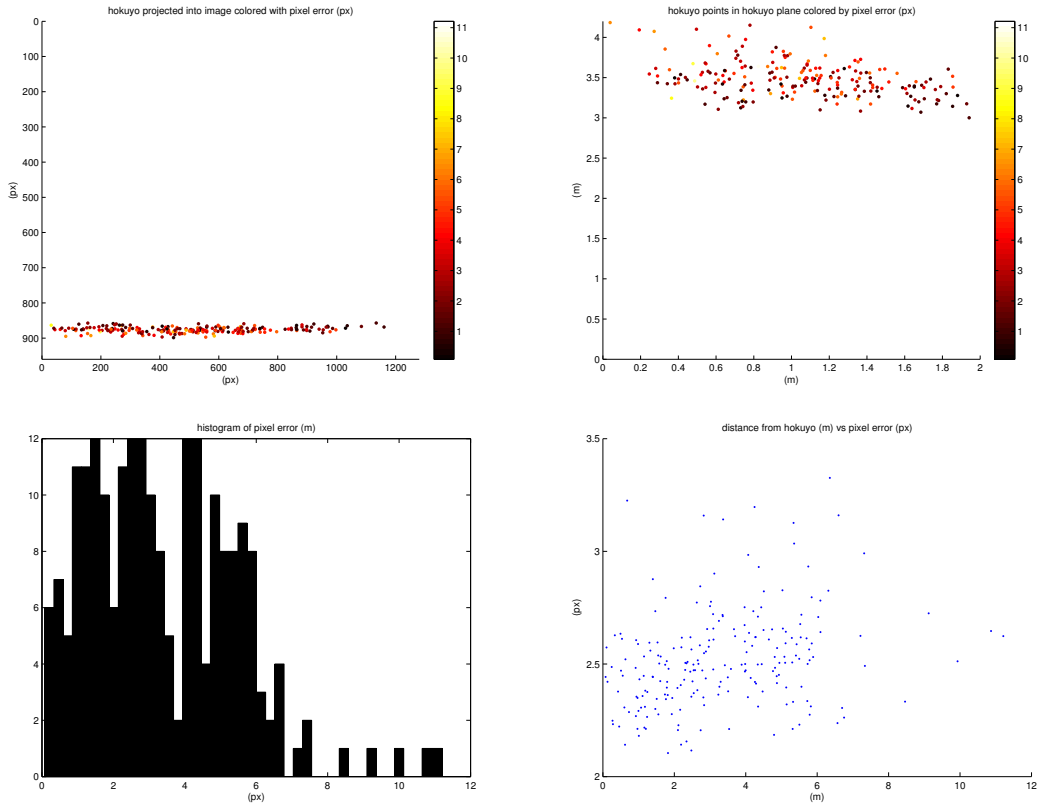

Figure 2.10: Error plots of how this calibration fits the underlying model. Clockwise from top left: a) All of the selected LiDAR points projected into an image plane and colored with their pixel error. b) All of the selected LiDAR points in their own plane colored with their pixel error. c) The distribution of LiDAR points vs pixel error. d) A histogram of total error in LiDAR points measured in pixel reprojection error.

Chapter 3

Global Representation for Registration

3.1 Introduction

In the past decade there has been an explosion of consumer depth and range sensors introduced into the market. The prices have fallen well into the consumer range which has necessitated algorithms for tasks like registration without expensive IMU's. In the same way that automatic image stitching is now a widely used tool, we expect that there will be a need for a fast and fully automatic solution for the range registration problem.

As with any registration problem, range registration consists of matching and estimating the rigid transformation. Because of the Iterative Closest Point Algorithm (ICP) section 1.2 we draw a distinction between rough and fine alignment. ICP is generally accepted as the gold standard for rigid fine alignment. Our algorithm thus solves the rough alignment problem. The rough alignment is difficult beacuse we must first determine the correspondence with no prior notion of how the pieces fit together.

In commercial products, initial alignment is achieved manually or by the use of characteristic markers in the scene. They further rely on a significant percentage of overlap between the two point sets. Another group of techniques depend on the extraction of local

features with such distinguishable attributes that correspondence becomes a non-iterative task. Robust variations like RANSAC reject outliers and improve the estimation of the rigid transformation.

In this chapter, the emphasis is put on the crude alignment step, which presents the real challenge for practical applications. We pose the following requirements for a registration algorithm: Fully automatic without artificial landmarks, partial overlap of point sets, independence of sensors and their sampling density as well as the size of the environment, and real-time, meaning no dependence on convergence speed.

Our algorithm is global and does not necessitate any feature detection. Its novel contribution is the reliable estimation of orientation between two Extended Gaussian Images (EGI) [Horn, 1984]. Our rotation estimate is obtained by exhaustively traversing the space of rotations to find the one which maximizes the correlation between EGIs. Such a computation would seem grueling, but we show how such a correlation can be computed efficiently using the spherical harmonics of the EGI and the rotational Fourier transform. To rotationally align point clouds with low overlap, we introduce a new representation of the EGI which we call the constellation image. This image captures the critical orientation distributions of a point cloud and can be correlated to obtain alignment without being adversely affected by outlying normal densities. We use a correlation-based formulation to subsequently estimate the translation. Our experiments show that our algorithm aligns point clouds arising from small objects or indoor scenes with as low as 45% overlap.

3.2 EGI and Orientation Histograms

Global representations of range scans are desirable because they capture characteristics which encode invariance and allow for direct comparisons for alignment and recognition tasks. Surface orientation histograms are effective approximations to the EGI representa-

tion, and throughout this text we will refer to the EGI and the orientation histogram interchangeably. Although it may seem like a simple accumulation of surface normals, the EGI provides a very powerful representation that allows for the direct recovery of orientation independent of any translational shift present.

Let us begin by defining precisely what an Extended Gaussian Image is. First, consider an object and a small region on that object which we will call δO. We consider the normal to this patch as a smoothly varying function where each point on the surface of the object maps to a point on the unit sphere, so for any region δO we can define a corresponding region δS on the sphere made up of the normals present in the δO region. For planes, this will map to a point on the sphere, but for smooth regions they will map to another region on the sphere. Now define the Gaussian Curvature

$$K = \lim_{\delta O \to 0} \frac{\delta S}{\delta O} = \frac{dS}{dO}$$

as the ratio of the size of the region on the gaussian sphere to the size of the corresponding region on the object. Now we can look at the following two integrals:

$$\iint_O K dO = \iint_S dS = S,$$
$$\iint_S \frac{1}{K} dS = \iint_O dO = O,$$

which necessitates using the inverse Gaussian Curvature in defining the EGI, and explains why we use the mass of each point in computing the orientation histogram.

Estimating attitude via EGI alignment has been discussed as early as Brou [Winter 1984]. These methods usually involve identifying and matching local features. Since there is a unique EGI representation for any convex object [Smith, 1979], this may be sufficient when registering orientation histograms of convex objects with much overlap. However, when dealing with range scans with low overlap, noisy measurements, or multiple discon-

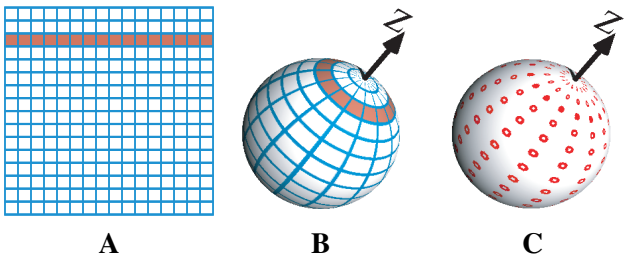

Figure 3.1: On the left (**A**) is a representation of an orientation histogram with 256 bins. The sphere $\mathbb{S}^2$ is sampled uniformly in spherical coordinates, creating a square grid. (**B**) depicts the corresponding bin sizes and shapes on the sphere. The highlighted bins correspond to the highlighted row in (**A**). (**C**) displays the bin centers when the longitudinal samples do not include the poles.

nected, nonconvex objects, it is unlikely that local feature generation and matching will be sufficient.

It has been shown in Makadia et al. [2004] that signal correlation provides a reliable measure for the rotational alignment of hemispherical images with little overlap. While such an evaluation would appear to require an expensive search, a fast correlation can be estimated using spherical Fourier analysis, with the requirement that our histogram bins be uniformly (in angular coordinates) spaced on the sphere.

Ideally, an orientation histogram would be comprised of bins which all have the same surface area and shape. One way to achieve this goal is by projecting regular polyhedra onto the sphere, but the regular polyhedron with the most sides is the icosahedron (twenty). At this scale, the histogram will not retain much distinguishing information. For finer sampling, approximations can be achieved easily by further subdividing the faces of the polyhedra. One purpose of retaining a constant bin shape and size is to provide a consistency for matching local features generated directly from the bin values. The cell shape and distribution we will use depends on the alternate criteria of a fast correlation. This will require uniform sampling in the spherical longitudinal and azimuthal coordinates. Figure 3.1 shows the effects of this choice on cell shapes and sizes on the sphere. As is clear from

the images, the bin sizes closest to the equator have the largest surface areas, and the bins closest to the north and south poles are the smallest. In fact, for a histogram with 256 bins (as pictured), the spherical surface area of the largest bin is roughly 10 times the surface area captured by the smallest bin.

3.3 Constellation EGI

One method for estimating rotations between EGI images is spherical correlation. Given two EGI images, we compute the rotation that maximizes the correlation between them (Details are shown in Makadia et al. [2006]). However, sometimes the non-overlapping region will have significant normals, in this case the higher peaks will correlate more and overpower the finer detail present in the EGI.

In order to get around this problem we propose extracting local maxima from the EGI and using them as feature points. The EGIs can then be aligned by finding the rotation that matches the most number of peaks. Originally this was done by constructing a binary EGI image and using spherical correlation, but later we found that it is simpler to extract the peaks directly from the data and align them using a Hough Tansform.

We define notation: a set of points $\mathbf{x}_i \in F_X$, normals $\mathbf{n}_i \in F_N$, and point denstities $d_i \in F_D$. Point densities can be easily computed by assuming locally uniform sampling and using a nearest neighbor library to efficiently find the nearest few neighors. If the function $\mathbf{p} = \text{KNN}(\mathbf{x}_i, k)$ is the function returning the kth nearest neighbor to $\mathbf{x}_i$, $\mathbf{p}$. Then we can compute the density at $\mathbf{x}_i$, to be

$$d_i = \frac{k}{\pi ||\mathbf{x}_i - \mathbf{p}||^2}.$$

We also define the normal density as

$$v_i = \sum_j \max(0, \tau_1 - \arccos(n_i^T n_j) d_j)$$

where τ_1 is a threshold which controls the amount of smoothing in the EGI for determining local maxima.

We define the set of peak indices C such that

$$C = \{i | v_i \geq v_j \; \forall \; n_j \in N_{\tau_2}(n_i)\}$$

where $N_{\tau_2}(n_i)$ is the neighborhood around n_i in the normal space of radius τ_2 (the neighborhood size for determining local maxima). This allows for multiple maxima right next to each other, but it is easy to remove these as the distance between maxima is easily checked.

To match these images we then enumerate all possible correspondences consisting of two peaks from each constellation set:

$$\begin{aligned} A(C_a, C_b) = \{(i,j,k,l) | & i \in C_a, j \in C_a, i \neq j, \\ & k \in C_b, l \in C_b, k \neq l, \\ & |\arccos(n_i^T n_j) - \arccos(n_k^T n_l)| < \tau_3, \\ & \tau_3 < |\arccos(n_i^T n_j)| < \pi - \tau_4, \\ & \tau_3 < |\arccos(n_k^T n_l)| < \pi - \tau_4\}, \end{aligned}$$

which says that we choose two points from each set, they can't be the same point, the angles between the pair must be similar in each set, and the angle between the pairs must not be close to zero or π. τ_3 and τ_4 are used to set the acceptable range between peaks. If the peaks are too close they won't give a reliable estimate. If the peaks are too close to one another ot too close to antipodal there will be an ambiguity in the estimated rotation. Then

we compute a rotation from each element of the enumeration by constructing a rotation matrix directly from the star locations. We define a reference frame via a rotation matrix for each pair of points:

$$R_{ij} = \begin{bmatrix} a^T \\ b^T \\ c^T \end{bmatrix}, \quad a = \frac{n_i + n_j}{||n_i + n_j||}, \quad c = \frac{(n_i - n_j) \times a}{||(n_i - n_j) \times a||}, \quad b = c \times a.$$

We can align these reference frames to get the rotation matrix R_{ijkl} of the correspondence that rotates n_k,n_l to align best with n_i,n_j

$$R_{ijkl} = R_{ij} R_{kl}^T.$$

Now that we have all of the votes for rotations, we must accumulate the votes in order to get a short list of possible rotations that is ready for translation estimation and verification. We select these in the same manner that we selected peaks in the EGI, by computing density in the voting (rotation) space and extracting the local maxima. However, $\mathbf{SO}(3)$ is not a euclidean space, so to get around this we use the same exponential map as described in Section 2.4. Recall that $R(\omega) = \exp\{\hat{\omega}\}$ maps ω to the open π ball, and the distances are more meaningful the closer to the center of approximation.

For rotations, we compute the local maxima slightly different:

$$a_i = \sum_j \max(0, \tau_5 - ||\omega_j - \omega_i||),$$

where a_i represents the vote density at vote location ω_i. ω_i and ω_j are two vote locations in the current tangent space, and τ_5 is a voting size. Then we extract the set of indices that are

local maxima:

$$B = \{i | a_i \geq a_j \; \forall \; \omega_j \in N_{\tau_5}(\omega_i)\}.$$

Finally, for each element in B we recompute all $\omega_j^{(i)}$ in the local tangent space to R_i and compute $B^{(i)}$:

$$a_i^{(i)} = \sum_j \max(0, \tau_5 - ||\omega_j^{(i)} - \omega_i^{(i)}||),$$

$$B^{(i)} = \{i | \omega_i^{(i)} < \tau_5, a_i \geq a_j \; \forall \; \omega_j^{(i)} \in N_{\tau_5}(\omega_i^{(i)})\}.$$

We next return all rotation matrices:

$$B_{final} = \{R_i | i \in \bigcup_i B^{(i)}\}.$$

This method is an exhaustive search over the space of corresponding peaks. It performs well if the number of peaks is kept small on the order of ten or twenty, but it does not scale to large numbers of peaks. We chose to carefully tune the τ smoothing parameters so that the number of peaks was kept resonable and larger features were used, but this could also be addressed instead by utilizing other methods of correspondence detection such as RANSAC or Geometric Hashing. Geometric Hashing may be especially interesting since it would allow for the hash to be simply the distance between points, and for each distance bin a pattern of peaks could be recorded. This would improve the scaling properties of the algorithm in the number of models being matched, it may improve the matching accuracy but that would depend on how well smaller support peaks perform in terms of repeatability when scanned from different perspectives.

3.4 Estimating the translation

Our use of the shift-invariant orientation histograms allowed us to decouple our alignment problem into consecutive searches for the rotational and translational components. We can formulate an estimate of the 3D translation which relies on the assumption that correct alignment is achieved at the locations of greatest overlap or correlation between range scans. We will define our range scans as a 3D histogram defined on a discrete set of bin locations in $\mathbb{R}^3$. First we define a set of centers C distributed over the pointcloud on a regular grid. Then we define the function $F(\mathbf{c}_i)$ to be the number of points in the pointcloud which are closest to this bin center $\mathbf{c}_i$:

$$F(\mathbf{c}_i) = \sum_{\mathbf{x}_j \in F_X} \mathbf{1}(||\mathbf{x}_j - \mathbf{c}_i|| < ||\mathbf{x}_j - \mathbf{c}_k|| \; \forall \; \mathbf{c}_k \in \{C/\mathbf{c}_i\}).$$

This can be efficiently computed using nearest neighbor libraries.

Applying our principles of correlation, we claim that the correct translational shift $\tau \in \mathbb{R}^3$ maximizes the following correlation function:

$$G(\tau) = \int_{x \in \mathbb{R}^3} F_1(x) F_2(x - \tau) dx. \tag{3.1}$$

Since (3.1) is a convolution integral, we know that the Fourier transform of $G(\tau)$ is given simply as

$$\hat{G}(k) = \hat{F}_1(k)\hat{F}_2(k).$$

The Fourier coefficients $\hat{F}_{\{1,2\}}(k)$ of the occupancy functions $F_{\{1,2\}}(x)$ can be recovered from the traditional $\mathbb{R}^3$ Fourier transform. In order for the correlation (3.1) to succeed, we must ensure overlap by generating a voxel space representation of $\mathbb{R}^3$ where each voxel covers a much larger area than the fine resolution of a range scanner.

Now that we have described the registration estimation, we will present the details of the verification step and recap the full algorithm.

3.5 Verification

In order to validate a hypothesized range alignment, we employ two different criteria. The first is based on the consistency of surface orientations in the overlapping regions of the aligned scans (the assumption is that normals should be the same for the points which overlap). If we voxelize the space after alignment, we can generate a global consistency measure by accumulating the difference in mean normal orientations for all overlapping voxels weighted by the mass of points present in each voxel:

$$V_{norm} = \sum_{\mathbf{c}_i \in C} \frac{F_1(\mathbf{c}_i) + F_2(\mathbf{c}_i)}{2} ||N_1(\mathbf{c}_i) \times N_2(\mathbf{c}_i)||,$$

where $F_1(\mathbf{c}_i), F_2(\mathbf{c}_i)$ are the number of points in bin $\mathbf{c}_i$ in each input pointcloud, and $N_1(\mathbf{c}_i), N_2(\mathbf{c}_i)$ represent the average normal bin $\mathbf{c}_i$ for each pointcloud. Figure 3.2 demonstrates how this test would be violated.

The second verification criteria we consider is visibility information. Intuitively, we would like to discard any alignment that would interfere with the line-of-sight of a range scanner. This method is similar to the visibility constraints explored in King et al. [2005], and figure 3.3 illustrates this test. Consider a point cloud F_2 being mapped into the reference frame of a point cloud F_1. If, after a hypothesized alignment, the surface in F_2 occupies the open space between the scanner viewpoint and surface of F_1, we claim that visibility of F_1 has been occluded and such an alignment is improbable.

More concretely, if $\mathbf{x}_i$ is a point in scan 1, and $\mathbf{x}_j$ is a point in scan 2. Next we define the point $\hat{\mathbf{x}}_i$ to be the point that scan 2 saw where it would have seen $\mathbf{x}_i$. We can get the distance to this point and the original point relative to scanner 2 which are defined $d_2(\hat{\mathbf{x}}_i)$

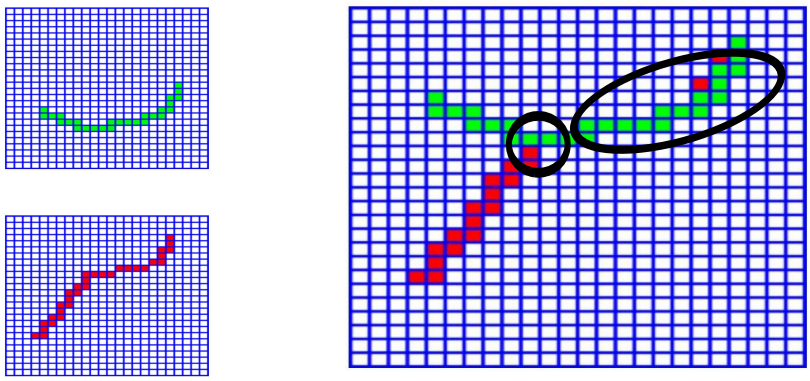

Figure 3.2: This cartoon illustration shows two input voxelized pointclouds on the left, and the combined image on the right. The left circle indicates a location where normals will disagree and the right circle shows a location where normals agree.

and $d_2(\mathbf{x}_i)$ respectively. The line-of-sight test is thus defined:

$$V_{LOS1} = \sum_{\mathbf{x}_i \in \{F_1 \cap F_2\}} H(d_2(\hat{\mathbf{x}}_i) - d_2(\mathbf{x}_i) - \gamma),$$

where γ is a parameter set to a value similar to the scanner noise, and $\mathbf{x}_i \in \{F_1 \cap F_2\}$ means that the point $\mathbf{x}_i$ which normally belongs to the pointcloud F_1 is only selected if the pointclouds 1 and 2 overlap at its position. Finally, $H(a)$ is the heaviside step function whos definition is reproduced here:

$$H(a) = \begin{cases} 1 & a \geq 0 \\ 0 & a < 0. \end{cases}$$

We can now reject any scan whos V_{norm} is below a threshold, and V_{LOS1} or V_{LOS2} are above a threshold.

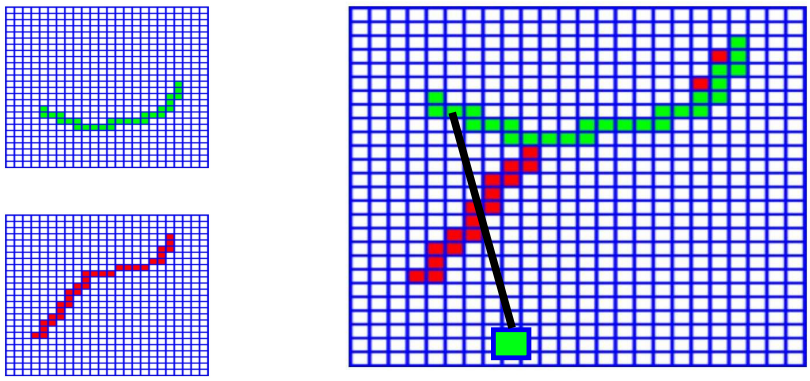

Figure 3.3: This cartoon illustration shows two input voxelized pointclouds on the left, and the combined image on the right. The line of sight violation is shown as a black line. Note that the red scan should have occluded the green scan from hte green sensor's point of view.

INPUT

1. Point Clouds $F_{X_1}, F_{X_2}, \ldots, F_{X_n}$.

ONLINE

1. Compute surface normal fields $F_{N_{\{1,2,\ldots,n\}}}$ for point clouds $F_{X\{1,2,\ldots,n\}}$.
2. Compute point densities $F_{D_{\{1,2,\ldots,n\}}}$ for point clouds $F_{X\{1,2,\ldots,n\}}$.
3. Generate Constellation EGI's $C_{\{1,2,\ldots,n\}}$ from normal fields and densities $F_{N_{\{1,2,\ldots,n\}}}$, $F_{D_{\{1,2,\ldots,n\}}}$ (section 3.3).
4. To estimate alignment between any two point sets F_{X_i}, F_{X_j}:
   (a) Compute set of rotation candidates $B_{final_{ij}}$ (section 3.3).
   (b) For each candidate rotation, estimate the translation by correlating the rotationally aligned scans (section 3.4).
   (c) Accept all aligned scans that pass the verification step (section 3.5)
5. Repeat until a cycle is found through all scans $F_{X_{\{1,2,\ldots,n\}}}$.
6. Obtain fine registration with pairwise ICP.

Figure 3.4: An outline of the automated point-cloud registration algorithm.

3.6 Experimental results

We now present the experimental results of our fully automated alignment algorithm as described in figure 3.4. The first step is to estimate surface normals, which can be obtained by computing the spherical gradient directly on a spherical depth-map. If such a representation is not available, then a simple local plane fitting approach can be used. To estimate the translational component of alignment, our correlation-based approach requires a voxelization of the point space. We chose our voxel size to roughly generate a voxel space of no more than 100 bins in any dimension. A fine registration from the estimated crude alignment is obtained with the Scanalyze ICP software freely available at http://graphics.stanford.edu/software/scanalyze/.

For many of our experiments we used spherical harmonics directly instead of the more accurate constellation exhaustive search. For these results we generated EGIs by choosing a signal bandwidth of $L = 128$, corresponding to a spherical histogram with 256×256 bins. We begin our evaluation with scans of the Happy Buddha provided by Curless and Levoy [1996]. Figure 3.5 shows the results of our estimation algorithm for a total of ten scans. Since this data was originally captured to test with ICP, the initial displacements are not very large. To test our approach, we applied random transformations to the starting point sets to create a scenario where a direct ICP would fail. The crude alignment is quite sufficient to initialize the fine estimation.

The second set of scans tested were of a statue of a lion. The scale is a bit larger than the Buddha model, nearing nine feet in height. The scans were captured using the DeltaSphere-3000 laser scanner. Fifteen scans circling the lion were taken. In Figure 3.6, some overall drift is apparent in the crude alignment. However, the quality of the pairwise matching is sufficient for ICP to converge correctly.

Our final evaluation deals with scans at yet another scale. Figure 3.7 depicts four scans

of a room taken with the DeltaSphere^{TM}-3000. The room was over 170 ft^2 in area (the volume was 1600 ft^3). A fifth scan capturing the entire room was also taken, allowing us to compare our rough alignment to a ground truth measurement. The crude alignment works well for each scan and a subsequent fine registration yields a very tight solution. One pair of successfully matched scans had only a 45% content overlap. Due to the dominant planes present in each scan, constellation images were needed before the correct alignment could be found and verified. Fewer than ten hypotheses were tested before a valid alignment was recovered in each case. We estimated a median error of 1.2 inches (computed as the min distance from every point in the rough aligned scan to the full scan). We also recorded the motion estimates for each pairwise ICP in the final alignment. The mean rotation angle was $2.0°$ (with a max of $2.1°$) and the mean translation was 3.8 inches (with a max of 5.0 inches).

3.7 Conclusion

We have presented a comprehensive algorithm for the automatic alignment of 3D point clouds designed specifically for multiple scans with little overlap. The correlation alignment of orientation histograms and constellation images is performed efficiently by extending the convolution theorem to spherical correlation. These methods, along with a reliable verification scheme, provide a crude alignment that yields a quality initialization for fine alignment. The crude alignment performs equally well without modification on small scale scans of models as well as large scale point clouds obtained with room scanners. The authors of Osteen et al. [2012] produced an implementation of our algorithm for use on a mobile robot with a Microsoft Kinect Sensor which shows that this method is still applicable to modern scanners.

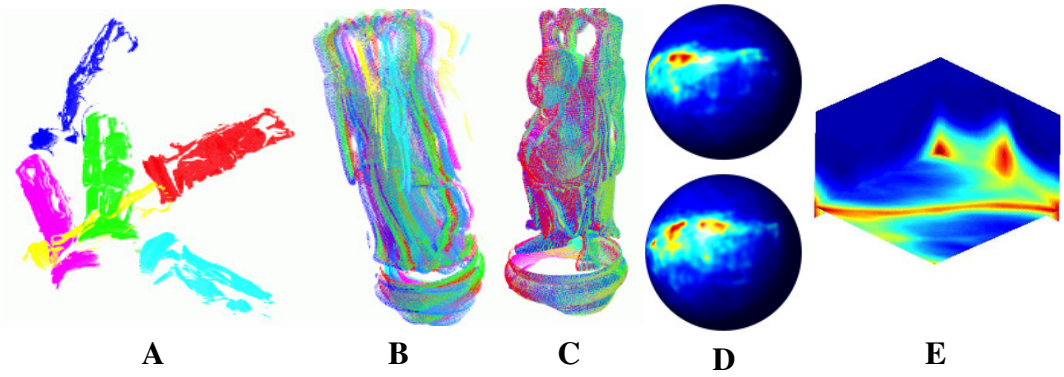

Figure 3.5: Registration of the Happy Buddha. **(A)** shows a the initial positions of some representative scans. **(B)** shows the rough alignment of ten point sets. **(C)** shows the final alignment for all scans after ICP is run after the crude registration. **(D)** shows a pair of EGIs from two of the scans, and **(E)** shows a slice of the correlation grid $G(R)$ at the location of the estimated rotation.

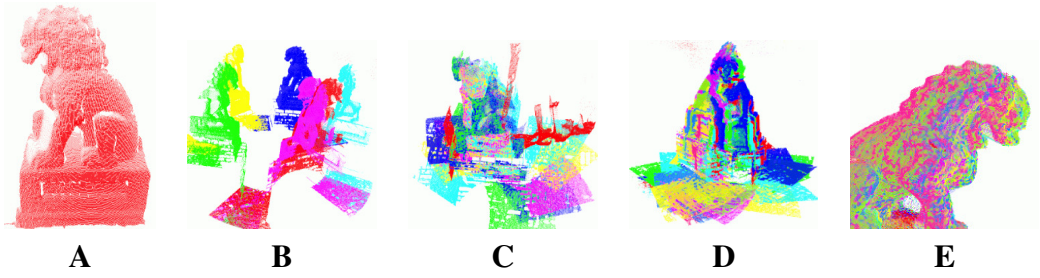

Figure 3.6: Registration of scans of a lion statue. **(A)** is a representative scan depicting the structure of the statue. **(B)** shows 6 scans in their initial positions. **(C)** shows the failure of running ICP directly on the input scans. **(D)** depicts the rough alignment. **(E)** shows one view of the successful final registration of all 15 scans.

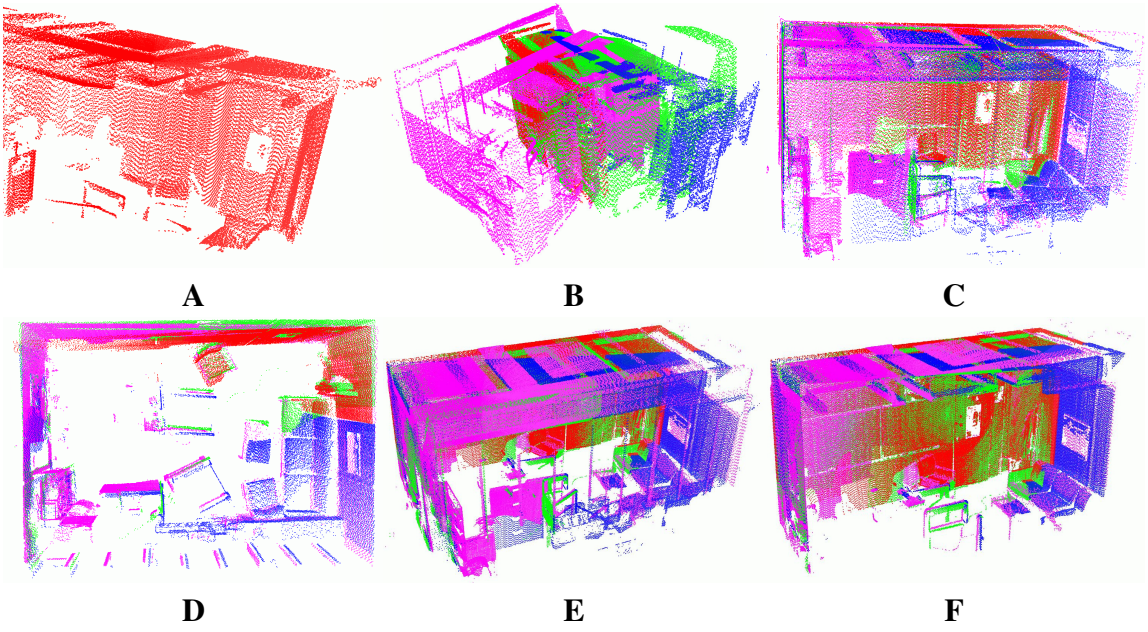

Figure 3.7: (**A**) shows a representative room scan. (**B**) shows the poor alignment obtained by running ICP on the input. (**C**, **D**) show a side and overhead view of the rough alignment. (**E**, **F**) show a full and partial view of the final alignment.

Chapter 4

Object Detection from Large-Scale 3D Datasets using Bottom-up and Top-down Descriptors

4.1 Introduction

Object detection and recognition in images or videos is typically done based on color and texture properties. This paradigm is very effective for objects with characteristic appearance, such as a stop sign or the wheel of a car. There are, however, classes of objects for which 3D shape and not appearance is the most salient feature. Cars are an object category, whose appearance varies a lot within the class, as well as with viewpoint and illumination changes. Instead of representing these objects with a collection of appearance models, specific to each viewpoint, several researchers have used range scanners and addressed object recognition in 3D. Range as an input modality offers the advantages of not losing a dimension or scale due to projection. In addition, figure-ground segmentation is easier in 3D than in 2D images since separation in depth provides powerful additional cues. On the

Figure 4.1: Cars detection results from real LiDAR data. Cars have been colored randomly.

other hand, range sensors have significantly lower resolution compared to modern cameras and alignment between the query and the database models still has to be estimated. The challenges associated with object detection in 3D are due to intra-class shape variations; different sampling patterns due to different sensors or a different distance and angle between the sensor and the object; targets that are almost always partial due to self-occlusion and occlusion; clutter; and computational efficiency.

In this chapter, we present an approach for detecting and recognizing objects characterized by 3D shape from large-scale datasets. The input is a point cloud acquired by range sensors mounted on moving vehicles. A part of the input is used as training data to provide manually labeled exemplars of the objects of interest, as well as negative exemplars where objects of interest are not present. Our algorithm automatically detects potential locations for the target objects in a bottom-up fashion. These locations are then processed by the top-down module that verifies the hypothesized objects by aligning them with models from the training dataset. We show results on a very large-scale dataset which consists of hundreds of millions of points. To the best of our knowledge, no results have been published for datasets of this size. State-of-the-art 3D recognition systems on real data [Carmichael

et al., 1999; Johnson et al., 1998; Matei et al., 2006; Mian et al., 2006; Ruiz Correa et al., 2006] have shown very high recognition rates, but on high-resolution scenes containing a few small objects captured in controlled environments.

We believe that our research is a first step towards fully automatic annotation of large scenes. Recent advances in sensor technology have made acquisition and geo-registration of data possible. Detailed 3D models can be generated very efficiently to provide high-quality visualization [Frueh et al., 2005], but their usefulness for everyday applications is limited due to the absence of semantic annotation. Much like image-based visualizations, such as the Google Street View, these representations cannot answer practical questions, such as "where is the nearest gas station, mailbox or phonebooth". Automatic methods for scene annotation would dramatically increase the benefits users can derive from these large collections of data. While our methods are not currently capable of addressing the problem in its full extent, this paper introduces a framework for object detection from range data that makes a step towards automatic scene annotation. Some results on car detection can be seen in Fig. 4.1.

The main technical contribution of our work is the combination of a bottom-up and a top-down process to efficiently detect and verify the objects of interest. We use spin images [Johnson and Hebert, 1999] as local descriptors to differentiate between the target objects and clutter and Extended Gaussian Images (EGIs) [Horn, 1984] to ascertain the presence of a target at the hypothesized locations. This scheme enables us to process very large datasets with high precision and recall. Training requires little effort, since the user has to click one point in each target object, which is then automatically segmented from the scene. The remaining points are used as negative examples. Spin images are computed on both positive and negative examples. EGIs only need to be computed for the positive exemplars of the training set, since we never need to align a candidate object with clutter. Accurate alignment estimates between similar but not identical objects enable us to segment the

target objects from the clutter.

4.2 Algorithm Overview

Our algorithm operates on 3D point clouds and entails a bottom-up and a top-down module. The steps for annotation and training are the following:

1. Selects one point on each target object.
2. Extract selected target objects automatically from the background.
3. Compute surface normals for all points[1] in both objects and background.
4. Compute spin images on a subset of the points for both objects and background and insert into spin image database DB_{SI} (Section 4.3).
5. Compute an EGI for each object (not for the background). Compute constellation EGI and density approximation. Insert into EGI database DB_{EGI} (Section 4.4).

Processing on test data is performed as follows:

1. Compute normals for all points and spin images on a subset of the points.
2. Classify spin images as positive (object) or negative (background) according to their nearest neighbors in DB_{SI}.
3. Extract connected components of neighboring positive spin images. Each connected component is a query (object hypothesis).
4. Compute an EGI and the corresponding constellation EGI for each query.
5. For each query and model in DB_{EGI} (Section 4.4):

[1] During normal computation, we also estimate the reliability of the normals, which is used to select reference points for the spin images

(a) Compute rotation hypothesis using constellation EGIs.

(b) For each rotation hypothesis with low distance according to Section (4.4.3), compute translation in frequency domain.

(c) Calculate the overlap between query and model.

6. If the overlap is above the threshold, declare positive detection (Section 4.4).

7. Label all points that overlap with all the models of DB_{EGI} after alignment as object points to obtain segmentation.

4.3 Bottom-Up Detection

The goal of the bottom-up module is to detect potential target locations in the point cloud with a bias towards high recall to minimize false negatives (missed detections). Since detection has to be performed on very large point clouds, we need a representation that can be computed and compared efficiently. To this end we use spin images [Johnson and Hebert, 1999], which arguably are the most popular local 3D shape descriptors. A spin image is computed in a cylindrical coordinate system, defined by a reference point and its normal, by calculating the histogram of radial and elevation distances of the point's neighbors. The resulting histogram is 2D, and can be viewed as an image which spins around the normal of the reference point to accumulate points in its bins; hence the name spin image. An example of spin image computation can be seen in figure 4.2. Due to integration around the normal of the reference point, spin images are invariant to rigid transformations and can be matched by comparing corresponding bins. This is not the case with 3D shape contexts [Frome et al., 2004] or EGIs (Section 4.4) for which several rotation hypotheses have to be evaluated to determine a match. The computational cost of computing several rotation hypotheses per comparison is prohibitive for us. Therefore,

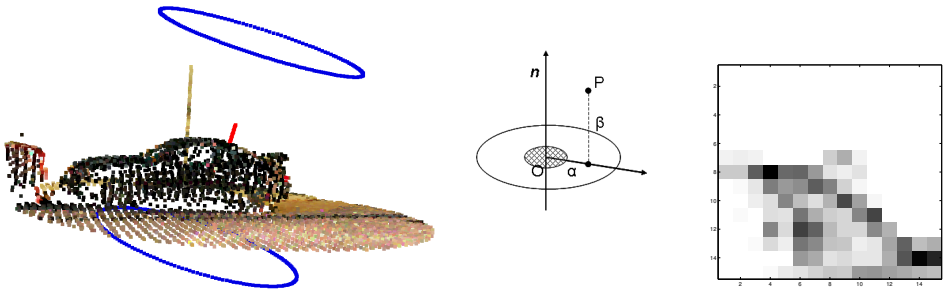

Figure 4.2: Left: spin image computation on real data. The blue circles are the bases of the cylindrical support region and the red vector is the normal at the reference point. Middle: illustration of spin image computation. O is the reference point and $\vec{n}$ its normal. A spin image is a histogram of points that fall into radial (α) and elevation (β) bins. Right: the spin image computed for the point on the car.

we use spin images, which are less discriminative but computationally cheaper, to detect targets.

Johnson and Hebert [1999] computed spin images on meshes. This can compensate for undesired effects due to varying sample density, since triangles contribute to each bin of the histogram with their area. Triangulating the point cloud to obtain a mesh is not trivial in our case, not only because of the computational cost, but also due to noise, sparsity and sampling patterns of the data. Similarly to Frome et al. [2004], we compute spin images directly from the point clouds and weigh the contribution of each point by its inverse density to account for sampling differences. Local density is computed in balls centered at every point. Isolated points are removed. Accounting for variations in point density is important for point clouds captured by range sensors since the density of samples on a surface is a function of sensor type, as well as distance and angle to the sensor.

Given a point cloud, regardless of whether it contains training or test data, normals for all points are computed using tensor voting [Medioni et al., 2000]. We then need to select reference points for the spin images. Our experiments have shown that spin images vary smoothly as long as the reference point is on the same surface and the normal is accurate.

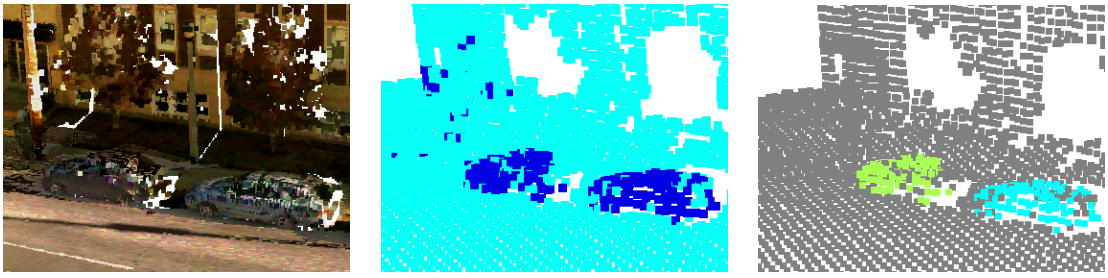

Figure 4.3: Left: input point cloud. Middle: Classification of spin images as target (blue) and background (cyan). (Only the reference points are shown.) Right: target spin image centers clustered into object hypotheses. Isolated target spin images are rejected.**Best viewed in color**

Therefore, reference points need to be dense enough to capture all surfaces of the object, but higher density is redundant. For cars, a distance of $0.4m$ was found to offer a good trade-off between coverage and computational efficiency. We obtain such a sampling by placing a 3D grid of the desired resolution in the dataset and dropping vertices that have no scanned points in their voxel. Since the reference points need to be among the points sampled by the scanner, the retained vertices are moved to the median of the nearest points. The grid can be seen in the two rightmost images of figure 4.3. A spin image is computed for each of these points unless the eigenvalues of the tensor after tensor voting indicate that the estimated normal is unreliable [Medioni et al., 2000]. Our spin images have 15 radial and 15 elevation bins resulting in a 225-D descriptor.

For the training data, the user has to specify the targets, which are assumed to be compact objects lying on the ground, by clicking one point on each. Then, an automatic algorithm segments the object as a connected component protruding from the ground. The ground can be reliably estimated in a small neighborhood around the selected point as the lowest smooth surface that bounds the data. Spin images computed for points on the targets are inserted into the spin image database DB_{SI} as positive exemplars, while spins images from the background are inserted as negative exemplars. We have implemented the database using the Approximate Nearest Neighbor (ANN) k-d tree [Arya et al., 1998].

During testing, query spin images are computed at reference points on a grid placed on the test data as above. Each query spin image is classified according to the nearest neighbor retrieved from DB_{SI}. Some results on real data can be seen in figure 4.3. Potential locations of the target objects can be hypothesized in areas of high density of positive detections, while isolated false positives can be easily pruned from the set of detections.

Object hypotheses (queries) are triggered by spin images that have been classified as positive (target). Target spin images are grouped into clusters by a simple region growing algorithm that starts from a spin image reference point and connects it to all neighboring target spin images within a small radius. When the current cluster cannot be extended any further, the algorithm initializes a new cluster. Raw points that are within a small distance from a cluster of spin images are also added to it to form a query. Since neighboring spin images overlap, our algorithm is robust to some miss-classifications.

4.4 Top-Down Alignment and Verification

The second stage of processing operates on the queries (clustered points with normals) proposed by the bottom-up stage and verifies whether targets exist at those locations. Spin images without geometric constraints are not discriminative enough to determine the presence of a target with high confidence. Spin image classification is very efficient, but only provides local evidence for the presence of a potential part of a target and not for a configuration of parts consistent with a target. For instance a row of newspaper boxes can give rise to a number of spin images that are also found in cars, but cannot support a configuration of those spin images that is consistent with a car. The top-down stage enforces these global configuration constraints by computing an alignment between the query and the database models using EGI descriptors.

Early research has shown that there is a unique EGI representation for any convex object

[Smith, 1979], which can be obtained by representing the surface normals of all points as unit vectors on a sphere. If the object is not convex, its shape cannot be completely recovered from the EGI, but the latter is still a powerful shape descriptor. The EGI does not require a reference point since the relative positions of the points are not captured in the representation. This property makes EGIs effective descriptors for our data in which a reference point cannot be selected with guaranteed repeatability due to occlusion, but the distribution of normals is fairly stable for a class of objects.

4.4.1 Computing EGIs

EGIs are computed for the positive object examples in the training set. Objects are segmented with assistance from the user, as described in Section 4.3. For the test data, an EGI is computed for each object hypothesis extracted according to the last paragraph of Section 4.3. Each EGI contains the normals of all input points of the cluster, oriented so that they point outwards, towards the scanner. These orientations can be computed since the trajectory of the sensor is available to us. The majority of objects are scanned only from one side and, as a result, the normals typically occupy at most a hemisphere of the EGI. This viewpoint dependance, however, occurs for both the queries and database objects and thus requires no special treatment. If necessary, database models can be mirrored to increase the size of the database without additional manual labeling since model symmetry is modeled by the EGI.

4.4.2 Constellation EGIs

Unlike spin images, comparing two EGIs requires estimating a rotation that aligns them before a distance can be computed. One can compute the rotational Fourier transform [Driscoll and Healy, 1994] to efficiently compute all correlations between EGIs [Maka-

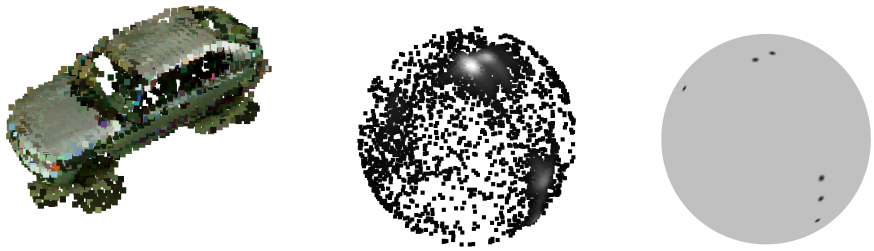

Figure 4.4: Left: a database model of a car. Middle: illustration of an EGI in which points are color-coded according to their density. Right: the corresponding constellation EGI.

dia et al., 2006]. This technique is efficient if all rotations need to be computed, but it is sensitive to clutter, missing parts and quantization. Our experiments have shown that quantization can have adverse effects on rotation and distance computations using EGIs. We can use the constellation EGI to cue a more efficient distance computation. Therefore, we avoid quantizing the orientations of the normals in an EGI and do not treat it as an orientation histogram.

Instead of an exhaustive search using either spatial or Fourier methods, we use a technique that generates discrete alignment hypotheses originally proposed in Makadia et al. [2006]. A *constellation EGI* records the locations of local maxima in the distribution of normals in the EGI. We call these maxima *stars*, since they resemble stars in the sky. An EGI and the corresponding constellation EGI for an object can be seen in Fig. 4.4. Two constellation EGIs can be matched by sampling pairs of stars that subtend the same angle on the sphere. Each sample generates matching hypotheses with two stars of the other EGI. If the angles between each pair are large enough and similar, a rotation hypothesis for the entire descriptor is generated. Note that a correspondence between two pairs of stars produces two possible rotations. Similar rotations can be clustered to reduce the number of hypotheses that need to be tested. The resulting set of rotations are evaluated based on the distance between the entire EGIs and not just the stars.

4.4.3 Hypothesis Verification

Conceptually, the rotation hypothesis that achieves the best alignment of the descriptors is the one that maximizes the cross-correlation between all normal vectors of the first and the second EGI. This computation is exact, but computationally expensive since models and queries consist of thousands of points each. To reduce the computational complexity, we select a smaller set of normals and compute the weights of kernels which are centered on this set, thus closely approximating the original EGI via interpolation. This computation is performed once per EGI and significantly reduces the cost of distance computation. Specifically, to create an approximation of the EGI for a set of input normals, we compute the density at all input normals on the sphere. We then select a subset of samples by greedily choosing a predetermined number of points. Each choice is made by computing the current interpolation via nearest neighbor, and then adding the normal with the largest deviation between approximated and actual values. Our method is similar to Carr et al. [2001], but operates on the sphere. Once we have a set of kernel centers N_s which is a subset of all normals N, the weights of the kernels are computed as follows:

$$v_{ij} = \frac{max(d_{max} - \arccos(\hat{n}_i^T \hat{n}_j), 0)}{\sum_j (max(d_{max} - \arccos(\hat{n}_i^T \hat{n}_j), 0))}, \quad i \in N, j \in N_s$$

$$D_j = V^\dagger D_i, \tag{4.1}$$

where D_j are the coefficients at the sparse set of normals N_s, and d_{max} is the range of the kernel function. Using this new representation, we can compute the distance between two EGIs, using a sparse set of samples, after applying a rotation hypothesis. If the two shapes are identical, the density values should be equal over the entire sphere. We measure the deviation from an ideal match by predicting the density on the samples of one EGI using the interpolation function of the other EGI and comparing them with the original density

values. Specifically we use the l_1 distance computed at the query points which we can now interpolate once the normals N_s are rotated according to each hypothesized rotation.

Let $d_1(\cdot)$ denote the interpolated density estimate on the first EGI and let $\hat{n}_i$ be the sparse set of normals of the first EGI. Let $\mathbf{R}_j$ be the rotation matrices corresponding to the set of hypotheses and $d_2(\cdot)$ be the interpolated density estimate on the second EGI. We want to select the rotation that minimizes the following distance function:

$$D_j = \sum_i |d_1(\hat{n}_i) - d_2(\mathbf{R}_j \hat{n}_i)|. \tag{4.2}$$

The minimum distance provides an estimate of the best rotation to align the two objects, but no estimate of translation and most importantly no indication of whether the objects actually match. For instance the "best" rotation between a building and a car could be one that aligns a wall with the side of the car. The final distance metric which allows us to make judgments on the similarity of two objects is presented in the next subsection. Typically, 1-5 rotations are close enough to the minimum distance. For these, we estimate the translation and compute the final distance in the following section.

4.4.4 Alignment and Distance Computation

Given the few best rotation hypotheses based in section 4.4.3, we compute the translation that best aligns the two models in the frequency domain. We adopt the translation estimation method of Makadia et al. [2006] in which translation is estimated using a Fourier transform in $\mathbb{R}^3$. This is less sensitive to noise in the form of missing parts or clutter than global alignment methods that estimate complete rigid transformations in the Fourier domain. We begin by voxelizing the model and the query to obtain binary occupancy functions in 3D. $F_1(x)$ and $F_2(x)$ are the voxel occupancy indicator functions for the model and the query, respectively. The optimal translation τ is the one that maximizes the following

function:

$$G(\tau) = \int_{x \in \mathbb{R}^3} F_1(x) F_2(x - \tau) dx. \tag{4.3}$$

The convolution can be re-written as multiplication in the frequency domain:

$$\hat{G}(k) = \hat{F}_1(k) \hat{F}_2(k), \tag{4.4}$$

where $\hat{F}_{\{1,2\}}(k)$ are the Fourier coefficients of the occupancy indicator functions $F_{\{1,2\}}(x)$.

Finally, we need a measure of distance to characterize the quality of the alignment that is flexible enough to allow for deformation between the query and the model. We experimented with the ICP distance [Besl and McKay, 1992], without performing ICP iterations, but found the overlap between the query and model to be more effective because the quantization in the translation estimation caused large ICP distance errors even though the models were similar. The overlap is computed as the inlier ratio over all points of the model and query, where an inlier is a point with a neighboring point from the other model that is closer than a distance threshold and whose normal is similar to that of the point under consideration. Figure 4.5 shows an alignment between a query and a database object and their corresponding EGIs. Selecting the inliers after alignment results in precise segmentation of the object from the background.

4.5 Experimental Results

We processed very large-scale point clouds captured by a moving vehicle equipped with four range scanners and navigation sensors. The dataset consists of about 200 million points, 2.2 million of which were used for training. The training set included 17 cars which were selected as target objects. We compute 81,172 spin images for the training set (of

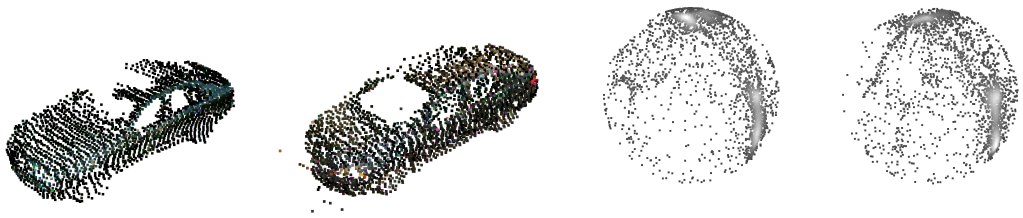

Figure 4.5: Alignment of a database model (left car and left EGI) and a query (right car and right EGI) that have been aligned. The car models are shown separately for clarity of the visualization. Notice the accuracy of the rotation estimation. The query has been segmented by the positive spin image clustering algorithm and the model by removing the ground after the user specified one point.

which 2657 are parts of cars) and 6.1 million for the test set. Each spin image has a 15×15 resolution computed in a cylindrical support region with height and radius both set to $2m$. Reference points for the spin images are selected as in Section 4.3 with an average distance between vertices of $0.4m$. The spin images of the training set are then inserted into DB_{SI}.

EGIs are computed for each target object in the training set, approximated by picking a smaller set of 200 normals, that minimize the interpolation error on all samples. The approximated EGIs are inserted into DB_{EGI}, which is a simple list with 17 entries. Since our method only requires very few representatives from each class, we were able to perform the experiments using a few sedans, SUVs and vans as models.

Spin images are computed for the test data and they are classified as positive or negative using a nearest neighbor classifier. Queries are generated by extracting connected components of positive spin images whose centers are within $1m$ of each other. This groups points roughly up to two grid positions away. EGIs are computed for the query objects and compared against model EGIs to compute the best alignment using the constellation model (Section 4.4.2). A hypothesis is generated by two pairs of stars, one from each EGI. Each pair must subtend an angle of at least 30° and the two angles must not differ by more than 5°. Rotations that meet these requirements are evaluated according to Section 4.4.3. For the best rotation hypotheses, the metric used to make the final decision is computed: the

Figure 4.6: Left: The precision-recall curve for car detection on 200 million points containing 1221 cars. (Precision is the x-axis and recall the y-axis.) Right: Screenshot of detected cars. Cars are in random colors and the background in original colors.

percentage of inliers on both models after alignment. For a point to be an inlier there has to be at least one other point from the other model that is within $30cm$ and whose normal deviates by at most 35° from the normal of the current point. We have found the inlier fraction to be more useful than the sum of squared distances.

Results on an test area comprising 220 million points and 1221 cars are shown in Figs. 4.6 and 4.7. The precision-recall curve as the inlier threshold varies is also shown in Fig. 4.6. For the point marked with a star, there are 905 true positives, 74 false positives and 316 false negatives (missed detections) for a precision of 92.4% and a recall of 74.1%.

4.6 Conclusion

We have presented an approach for object detection from 3D point clouds that is applicable to very large datasets and requires limited training efforts. Its effectiveness is due to the combination of bottom-up and top-down mechanisms to hypothesize and test locations of potential target objects. An application of our method on car detection has achieved very satisfactory precision and recall on an area far larger than the test area of any previously published method. Moreover, besides a high detection rate, we are able to accurately segment the objects of interest from the background. We are not aware of any other method-

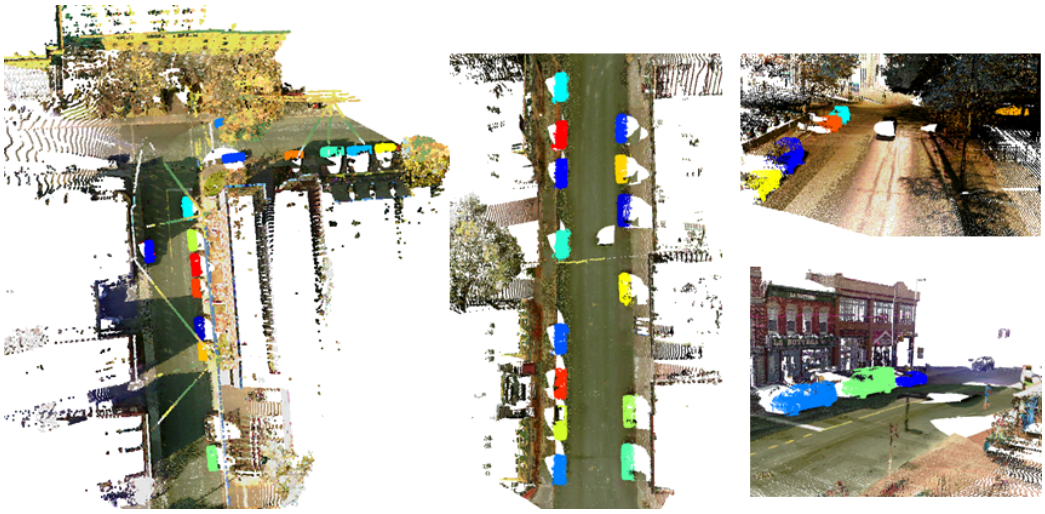

Figure 4.7: Screenshots of detected cars, including views from above. (There is false negative at the bottom of the left image.)

ology that obtains comparable segmentation accuracy without being trained on the same instances that are being segmented.

A limitation of our approach is that search is linear in the number of objects in the EGI database. We are able to achieve satisfactory results with a small database, but sublinear search is a necessary enhancement to our algorithm.

Chapter 5

Conclusions

We have presented a cohesive set of novel algorithms that are able to calibrate, align, and recognize objects in challenging 3D datasets. The LiDAR calibration and the CEGI alignment work are stable and polished enough that example code has been released.

For the calibration work presented in chapter 2, we demonstrated a fully automatic RANSAC system based on the minimal solution to calibrate a 2D line scan LiDAR system with a camera. The minimal solution could also be applicable to an online form of the algorithm where features in the environment could be detected and outliers rejected. The features used show that calibration can be done very close to the underlying points and that complicated and user intensive features such as chessboards are not required.

We also explained the intricacies of solving the least squares problem using Lie Algebra. Finally, we implemented a user interaction based calibration tool that can reliably produce accurate calibrations when employed by non-expert users and demonstrates the feasibility of using the point to line correspondences.

Accurate calibrations allow for the better fusing of LiDAR depth data with texture data from standard images, thus allowing more data sharing between modalities. Furthermore, calibration allows the use of other sensors to drive a motion based scanning system, such as

the Hokuyo camera system described, because the motion estimates could not be computed from the 2D LiDAR alone.

The range registration work presented in chapter 3 demonstrates, when combined with ICP for fine alignment and a reliable verification scheme, a robust method for aligning range scans. We demonstrated how to compute a representation of the Extended Gaussian Image, how to convert that image into our novel Constellation EGI, and how to align the CEGI images to produce rotational hypotheses. By decoupling the rotation and translation estimation steps we have greatly improved the running time of scan alignment. We demonstrated the effectiveness of our algorithm on a number of small scale scans and larger scans captured with a room scanner. Moreover, we demonstrated the usefulness of the CEGI as a descriptor by choosing it as our top down global descriptor for the recognition method of chapter 4. Other researchers are applying our method for rough registration using consumer depth cameras and other sensors, and we are routinely cited when a new range registration or 3D model estimation algorithm is proposed.

The city scale work of chapter 4 demonstrates an approach to object recognition which is applicable to very large datasets with minimal training effort. The effectiveness is due to the use of bottom-up features to drive attention, and top-down verification of detections. We learned what sorts of features perform well with the scanning parameters of the URGENT project. For example we learned what size of support is needed for spin images to be informative, and we explored some of the verification methods for confirming that an object matches a model. One limitation is that the search through the models is linear in the size of the database. We overcame this limitation by using a small database that generalized well, but in order to scale to larger databases a sublinear search would be necessary.

We believe that as robots and sensors try to perform more tasks they will need complicated algorithms and sensing modalities to gain the most understanding of their surroundings. Our experiences with building systems has taught us that many LiDAR tasks are

currently within the reach of engineering.

Bibliography

A. Adan, C. Cerrada, and V. Feliu. Global shape invariants: a solution for 3D free-form object discrimination/identification problem. *Pattern Recognition*, 34:1331–1348, 2001.

A. K. Aijazi, P. Checchin, and L. Trassoudaine. Segmentation based classification of 3d urban point clouds: A super-voxel based approach with evaluation. *Remote Sensing*, 5 (4):1624–1650, 2013.

A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. Rusu, S. Gedikli, and M. Vincze. Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. *Robotics Automation Magazine, IEEE*, 19(3): 80–91, 2012.

S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journ. of the ACM*, 45:891–923, 1998.

A. Ashbrook, R. Fisher, C. Robertson, and N. Werghi. Finding surface correspondence for object recognition and registration using pairwise geometric histograms. In *European Conf. on Computer Vision*, pages II: 674–686, 1998.

J. Behley, K. Kersting, D. Schulz, V. Steinhage, and A. Cremers. Learning to hash logistic regression for fast 3d scan point classification. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 5960–5965, 2010.

J. Behley, V. Steinhage, and A. Cremers. Performance of histogram descriptors for the classification of 3d laser range data in urban environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4391–4398, 2012.

J. A. Beraldin, L. Cournoyer, M. Rioux, F. Blais, S. F. El-Hakim, and G. Godin. Object model creation from multiple range images: acquisition, calibration, model building and verification. In *International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, 1997.

P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

J.-Y. Bouguet. Camera calibration toolbox for matlab. *www.vision.caltech.edu*, 2006.

P. Brou. Using the gaussian image to find the orientation of objects. *International Journal of Robotics Research*, 3(4):89–125, Winter 1984.

J. Canny. A computational approach to edge detection. *IEEE PAMI*, 8(6):679–698, 1986.

O. Carmichael, D. Huber, and M. Hebert. Large data sets and confusing scenes in 3-d surface matching and recognition. In *3DIM*, pages 358–367, 1999.

J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH*, pages 67–76, New York, NY, USA, 2001. ACM.

B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 303–312, New York, NY, USA, 1996. ACM.

H. Delingette, M. Hebert, and K. Ikeuchi. A spherical representation for the recognition of curved objects. In *International Conference on Computer Vision*, Berlin, 1993.

C. Dorai and A. Jain. Cosmos: A representation scheme for 3d free-form objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(10):1115–1130, October 1997.

J. Driscoll and D. Healy. Computing fourier transforms and convolutions on the 2-sphere. *Advances in Applied Mathematics*, 15:202–250, 1994.

M. Fischler and R. Bolles. Random sample consensus. *Communications of the ACM*, Jan 1981.

A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Proc. Eigth European Conference on Computer Vision*, pages 224–237, 2004.

C. Frueh, S. Jain, and A. Zakhor. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *International Journal of Computer Vision*, 61(2):159–184, February 2005.

T. Funkhouser and P. Shilane. Partial matching of 3d shapes with priority-driven search. In *Symposium on Geometry Processing*, 2006.

J. Gallier. *Notes on Differential Geometry and Lie Groups*. 2011.

J. Glover, R. Rusu, and G. Bradski. Monte carlo pose estimation with quaternion kernels and the bingham distribution. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

A. Halma, F. ter Haar, E. Bovenkamp, P. Eendebak, and A. van Eekeren. Single spin image-icp matching for efficient 3d object recognition. In *Proceedings of the ACM workshop on 3D object retrieval*, 3DOR '10, pages 21–26, New York, NY, USA, 2010. ACM.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

M. Hebert, K. Ikeuchi, and H. Delingette. A spherical representation for recognition of free-form surfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(7):681–690, 1995.

R. Horaud and F. Dornaika. Hand-eye calibration. *Intl. J. Robot. Res.*, 1995.

B. Horn. Extended gaussian images. *Proceedings of the IEEE*, 72(12):1656–1678, December 1984.

D. Huber and M. Hebert. Fully automatic registration of multiple 3D data sets. *Image and Vision Computing*, 21(7):637–650, 2003.

D. Huber, A. Kapuria, R. Donamukkala, and M. Hebert. Parts-based 3d object classification. In *Int. Conf on Computer Vision and Pattern Recognition*, pages II: 82–89, 2004.

K. Ikeuchi. Determining attitude of object from needle map using extended gaussian image. In *AIM-714: Tech Report*, MIT, 1983.

A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(5):433–449, May 1999.

A. Johnson, O. Carmichael, D. Huber, and M. Hebert. Toward a general 3-d matching engine: Multiple models, complex scenes, and efficient data filtering. In *Image Understanding Workshop*, pages 1097–1108, 1998.

A. E. Johnson and M. Hebert. Recognizing objects by matching oriented points. In *IEEE Conf. Computer Vision and Pattern Recognition*, Puerto Rico, June 17-19, 1997.

S. Kang and K. Ikeuchi. The complex egi: A new representation for 3-d pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(7):707–721, July 1993.

M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on Geometry Processing*, June 2003.

B. J. King, T. Malisiewicz, C. V. Stewart, and R. J. Radke. Registration of multiple range scans as a location recognition problem: Hypothesis generation, refinement and verification. In *Proceedings of the Fifth Intl. Conf. on 3D Digital Imaging and Modeling*, 2005.

J. J. Little. Determining object attitude from extended gaussian images. In *Proc. of the 9th International Joint Conference on Artificial Intelligence*, August 1985.

X. Liu, R. Sun, S. B. Kang, and H. Y. Shum. Directional histogram model for three-dimensional shape similarity. In *IEEE Conf. Computer Vision and Pattern Recognition*, Wisconsin, June 16-22, 2003.

Y. Liu, J. Pu, H. Zha, W. Liu, and Y. Uehara. Thickness histogram and statistical harmonic representation for 3D model retrieval. In *3DPVT*, Thessaloniki, 2004.

M. I. A. Lourakis and A. A. Argyros. Sba: A software package for generic sparse bundle adjustment. *ACM Trans. Math. Softw.*, 36(1):2:1–2:30, Mar. 2009.

L. Lucchese, G. Doretto, and G. M. Cortelazzo. A frequency domain technique for range data registration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(11):1468–1484, 2002.

J. Maciel and J. Costeira. A global solution to sparse correspondence problems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(2):187–199, 2003.

A. Makadia, L. Sorgi, and K. Daniilidis. Rotation estimation from spherical images. In *Proc. Int. Conf. on Pattern Recognition*, Cambridge, UK, 2004.

A. Makadia, A. Patterson, IV, and K. Daniilidis. Fully automatic registration of 3d point clouds. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1297–1304, 2006.

B. Matei, Y. Shan, H. Sawhney, Y. Tan, R. Kumar, D. Huber, and M. Hebert. Rapid object indexing using locality sensitive hashing and joint 3d-signature space estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(7):1111–1126, July 2006.

G. Medioni, M. Lee, and C. Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier, New York, NY, 2000.

C. Mei and P. Rives. Calibration between a central catadioptric camera and a laser range finder for robotic applications. In *ICRA*, pages 532 –537, may. 2006.

A. Mian, M. Bennamoun, and R. Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(10):1584–1601, October 2006.

F. Moosmann and M. Sauerland. Unsupervised discovery of object classes in 3d outdoor scenarios. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 1038–1044, 2011.

M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

O. Naroditsky, A. Patterson, IV, and K. Daniilidis. Automatic alignment of a camera with a line scan lidar system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3429–3434, 2011.

R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pages 127–136, 2011.

J. Novatnack and K. Nishino. Scale-dependent/invariant local 3d shape descriptors for fully automatic registration of multiple sets of range images. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision ECCV 2008*, volume 5304 of *Lecture Notes in Computer Science*, pages 440–453. Springer Berlin Heidelberg, 2008.

P. Nunez, P. Drews, R. Rocha, and J. Dias. Data fusion calibration for a 3d laser range finder and a camera using inertial data. *European Conference on Mobile Robots '09*, page 9, 2009.

R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Transactions on Graphics*, 21(4), 2002.

P. Osteen, J. Owens, and C. Kessens. Online egomotion estimation of rgb-d sensors using spherical harmonics. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1679–1684, 2012.

C. Papazov and D. Burschka. An efficient ransac for 3d object recognition in noisy and occluded scenes. In R. Kimmel, R. Klette, and A. Sugimoto, editors, *Computer Vision ACCV 2010*, volume 6492 of *Lecture Notes in Computer Science*, pages 135–148. Springer Berlin Heidelberg, 2011.

A. Patterson, IV, P. Mordohai, and K. Daniilidis. Object detection from large-scale 3d datasets using bottom-up and top-down descriptors. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision ECCV 2008*, volume 5305 of *Lecture Notes in Computer Science*, pages 553–566. Springer Berlin Heidelberg, 2008.

S. Ruiz Correa, L. Shapiro, M. Meila, G. Berson, M. Cunningham, and R. Sze. Symbolic signatures for deformable shapes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(1):75–90, January 2006.

S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proceedings of the Third Intl. Conf. on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

R. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *International Conference on Robotics and Automation*, pages 3212–3217, 2009.

R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3384–3391. IEEE, 2008.

R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE, 2010.

M. Saucy and D. Laurendau. a general surface approach to the integration of a set of range views. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(4):344–358, 1995.

D. Scaramuzza, A. Harati, and R. Siegwart. Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes. In *IROS 2007*, pages 4164–4169, 2007.

D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

A. Segal, D. Haehnel, and S. Thrun. Generalized-icp. In *Robotics: Science and Systems*, volume 2, page 4, 2009.

Y. Shan, H. Sawhney, B. Matei, and R. Kumar. Shapeme histogram projection and matching for partial object recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(4):568–577, April 2006.

D. A. Smith. Using enhanced spherical images. In *AIM-530: Tech Report*, MIT, 1979.

F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131–147, 1990.

I. Stamos and M. Leordeanu. Automated feature-based range registration of urban scenes of large scale. In *IEEE Conf. Computer Vision and Pattern Recognition*, Wisconsin, June 16-22, 2003.

I. Stamos, O. Hadjiliadis, H. Zhang, and T. Flynn. Online algorithms for classification of urban objects in 3d point clouds. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 332–339, 2012.

F. Stein and G. Medioni. Structural hashing: Efficient three dimensional object recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):125–145, February 1992.

C. J. Taylor and D. J. Kriegman. Minimization on the lie group so(3) and related manifolds.

Technical Report 9405, Center for Systems Science, Dept. of Electrical Engineering, Yale University, New Haven, CT, April 1994.

F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010*, pages 356–369. Springer, 2010a.

F. Tombari, S. Salti, and L. Di Stefano. Unique shape context for 3d data description. In *Proceedings of the ACM workshop on 3D object retrieval*, pages 57–62. ACM, 2010b.

R. Unnikrishnan and M. Hebert. Fast extrinsic calibration of a laser rangefinder to a camera. *Tech. Rep. CMU Robotics Institute*, page 339, 2005.

A. Velizhev, R. Shapovalov, and K. Schindler. Implicit shape models for object detection in 3d point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, I-3:179–184, 2012.

G. Weiß, C. Wetzler, and E. Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *IROS*, 1994.

X. Xiong, D. Munoz, J. Bagnell, and M. Hebert. 3-d scene analysis via sequenced predictions over points and regions. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2609–2616, 2011.

B. Yang, P. Sharma, and R. Nevatia. Vehicle detection from low quality aerial lidar data. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, pages 541–548, 2011.

Q. Zhang and R. Pless. Extrinsic calibration of a camera and laser range finder. In *IROS 2004*, volume 3, pages 2301 – 2306 vol.3, sep. 2004.