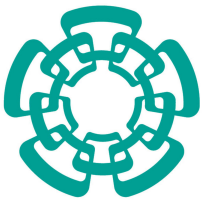


Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Escáner 3D de Alta Precisión

T E S I S

Que presenta

Daniel López Escogido

Para obtener el grado de

**Doctor en Ciencias
en Computación**

Director de la Tesis:
Dr. Luis Gerardo de la Fraga

Ciudad de México Junio, 2016

Resumen

En este trabajo se presenta el diseño y construcción de un escáner basado en un sensor láser 2D comercial de alta precisión. En el diseño propuesto se desplaza el objeto sobre el plano cartesiano xy, lo que posibilita obtener los puntos que describen la superficie de objetos del mundo real. Todos los puntos se guardan en archivos conocidos como archivos de *nubes de puntos*. Posteriormente se describe una metodología basada en el paradigma *Muestra Aleatoria de Consenso* (MAC, en inglés *Random Sample Consensus* (RANSAC)) y un muestreo no uniforme que permite ajustar los parámetros de primitivas geométricas a las nubes de puntos de una manera más eficiente que un muestreo uniforme y acelerar la convergencia del algoritmo MAC. En este trabajo los modelos que se proponen usar en el algoritmo implementado usan técnicas directas, o técnicas lineales, de estimación de primitivas geométricas basados en la descomposición de matrices en valores y vectores propios y sistemas de ecuaciones lineales, empleando solamente la información de las coordenadas de los puntos. Las primitivas geométricas que se trabajaron son: planos, esferas, cilindros y conos. También se hace la propuesta de la implementación de una metodología para el registro de nubes de puntos mediante el uso de descriptores de superficie, una combinación del algoritmo Puntos más Cercanos Iterativo (PMCI), (en inglés *Iterative Closest Point* (ICP)) y un método de mínimos cuadrados como lo es el algoritmo propuesto por Levenberg-Marquardt, dando como resultado una metodología útil para resolver el problema del registro en 3D de forma eficiente.

Abstract

In this work we present the design and build of a scanner based on a high precision commercial laser. In the proposed design, the scanned object is displaced over the xy Cartesian plane, which makes it possible to obtain all the points describing the surface object. All those scanned points are stored in a file named *cloud point file*. We also develop a methodology to extract geometric primitives inside the cloud point file. This methodology is based on the Random Sample Consensus (RANSAC)) paradigm. As we have the scanned points sorted in the scanned direction, we improve the execution time of RANSAC using a non-uniform sampling instead of the standard uniform sampling on unordered cloud point. In this work we use linear algorithms to estimate the consensus model of the geometric primitives plane, sphere, cylinder, and cone. These linear algorithms are estimated using the Singular Value Decomposition. Finally, we also use the Iterative Closest Point (ICP) algorithm to register two clouds of points, and we develop a new approach to use surface descriptors, instead of points, for the ICP algorithm. We use the standard non-linear optimizing algorithm Levenberg-Marquardt inside the ICP algorithm, for solving in an efficient way the problem of 3D registration.

Al único y sabio Dios, nuestro Salvador,
sea gloria y majestad, imperio y potencia,
ahora y por todos los siglos. Amén.
Jd. 25

Agradecimientos

En memoria del Dr. Adriano de Luca Pennachia, por haberme permitido trabajar con él y darme la oportunidad de ingresar al programa de doctorado bajo su dirección

Al Dr. Luis Gerardo de la Fraga por su dirección durante el desarrollo de este trabajo de investigación

A los revisores de este documento de tesis por sus comentarios: Dr. Carlos A. Coello Coello, Dr. Francisco Rodríguez Henrquez, Dra. Dolores Lara Cuevas y Dra. Luz Abril Torres Méndez

A todos los profesores del departamento de computación del CINVESTAV por compartir sus conocimientos conmigo

Al CINVESTAV por la oportunidad de desarrollarme personalmente en una institución de gran renombre

Al personal de secretarias por su apoyo en cada tramite necesario: Srta. Sofia Reza C, Srta. Felipa Rosas López y Srta. Erika B. Ríos Hernández

A mi familia por el apoyo y comprensión de toda la vida

Al CONACYT por el apoyo económico que me otorgó durante el tiempo de desarrollo de este trabajo de investigación

A los proyectos CONACYT 166763 y 168357 por el financiamiento parcial de este trabajo de investigación

Índice General

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Publicaciones

Congresos internacionales:

Daniel López-Escogido and Adriano De Luca. 2-D High Precision Laser Sensor for Detecting Small Defects in PCBs, 9th International Conference on Electrical Engineering, Computing Science and Automatic Control. IEEE Press, México City. 26-28 Sept. pages: 380-385, 2012

Daniel López-Escogido and Luis Gerardo de la Fraga, Automatic extraction of geometric models from 3D point cloud datasets, 11th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE). Cd. Del Carmen, Campeche, México. Oct. pages 1-5, 2014

Luis Gerardo de la Fraga and Daniel López-Escogido, Point cloud registration using surface descriptors, 10th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, 2-4 July, Funchal, Madeira, Portugal. 2016

Congresos nacionales:

Daniel López-Escogido and Luis Gerardo de la Fraga, An Implementation of an Algorithm for 3D Registration, X Taller-Escuela de Procesamiento de Imgenes (PI14), Centro de Investigación en Matemáticas A.C. (CIMAT), Guanajuato, Gto. 15 y 16 de octubre, 2014

Daniel López-Escogido and Luis Gerardo de la Fraga, Accelerating RANSAC for Matrix of 3D Ordered Points, XI Taller-Escuela de Procesamiento de Imágenes (PI15), Centro de Investigación en Matemáticas A.C. (CIMAT), Guanajuato, Gto. 5 y 6 de noviembre, 2015

1

Introducción

Para la digitalización de objetos del mundo real, es necesario el uso de sistemas o dispositivos que puedan sensar la forma superficial de los objetos y que estos dispositivos puedan devolver un conjunto de datos que describan tal superficie. La información generada por estos dispositivos comúnmente son un conjunto de datos desorganizados en un sistema de coordenadas xyz llamados $nubes\ de\ puntos$. A estos datos se les puede visualizar mediante herramientas de graficación en 3D, pero para que estos datos sean útiles se requiere de metodologías de análisis para extraer características de los objetos digitalizados, que permitan hacer una representación más precisa y eficiente con modelos matemáticos, además de permitir hacer una reconstrucción del objeto.

1.1 Antecedentes

La digitalización de objetos del mundo real tiene muchas aplicaciones y ha crecido la importancia de su uso en muchas áreas de la industria y la investigación como el modelado en CAD, ingeniería inversa, metrología, control de calidad, graficación por computadora,

vídeo juegos, visualización, visión por computadora, reconocimiento de patrones, robótica, etc. [22]. Por ejemplo, algunas publicaciones muestran el uso de la digitalización de objetos: en [35] se presenta un caso de ingeniería inversa para reconstruir partes mecánicas, en [54] los autores presentan el uso de escáners $LIDAR$ (por su nombre en inglés: *Light Detection And Ranging*) para su aplicación en la investigación en geología, en [70] Wulf *et al.* presentan un trabajo desarrollado en la aplicación de un escáner láser en la visión por computadora para detectar modelos geométricos para la navegación de un robot y en [43] se presenta el desarrollo de un escáner 3D para la adquisición de datos en áreas de trabajo.

1.1.1 Mesas de pruebas

Tanto en los laboratorios de pruebas, como en las áreas de trabajo existe la presencia de vibraciones y ruido que pueden producir errores en sistemas de medición de alta precisión. Estos tipos de vibraciones y ruidos del medio ambiente generalmente no son deseados en la puesta en funcionamiento de sistemas que ejecutarán labores de precisión. Estos tipos de señales físicas podrían ser aminoradas con el uso de mesas de prueba que disminuyan su efecto en el sistema. Estas mesas de prueba pueden combinar el uso de diferentes materiales y formas de construcción para evitar la interferencia del ruido y vibraciones en el funcionamiento del sistema. En la figura 1.1 se ilustra cómo el ruido y vibración del medio ambiente puede ser amortiguado al pasar de un medio A (que podría ser la tierra) a otro medio B de amortiguamiento.

1.1.2 Escáneres 3D

Un escáner 3D es un dispositivo que involucra la acción recíproca de diferentes componentes. Estos componentes deben permitir el sensado de diferentes partes de los objetos del mundo real para recolectar información que representa diferentes puntos de la superficie de los objetos.

Un escáner 3D puede ser catalogado de acuerdo a las diferentes tecnologías que em-

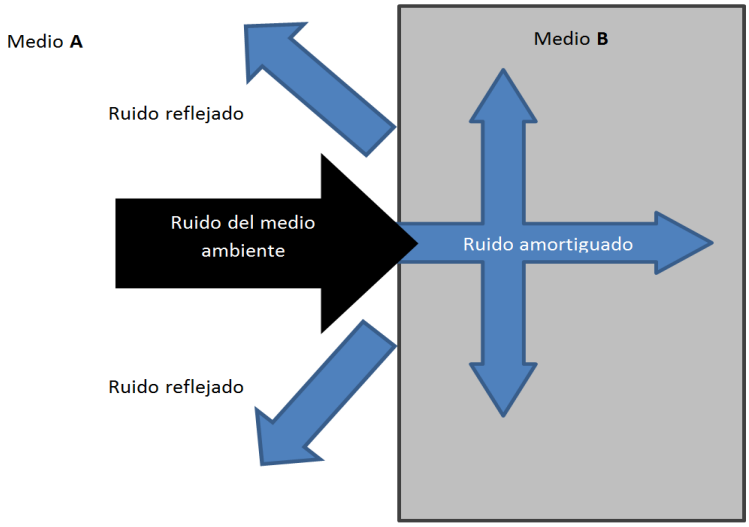


Figura 1.1: Ejemplo de absorción de ondas al pasar de un medio A a un medio B

plea, principalmente divididas en sensores con contacto y sensores sin contacto. Los escáneres sin contacto al mismo tiempo pueden ser divididos en escáneres activos y escáneres pasivos. En [10] se puede encontrar una clasificación de los escáneres 3D de acuerdo a la tecnología que emplean.

En la tecnología que usa sensores de contacto el dispositivo más común y preciso emplea Máquinas de Medición por Coordenadas (MMC) en combinación de una sonda de prueba [63, 67], en donde por cada coordenada de la máquina MMC, la sonda realiza una medida de la distancia de un punto de referencia hacia la superficie del objeto. Sin embargo, la lentitud de esta tecnología podría no ser adecuada para la digitalización de objetos de grandes dimensiones o incluso para objetos frágiles que podrían ser destruidos al momento de realizar la digitalización.

Los escáneres activos sin contacto emplean algún tipo de radiación como: ondas de luz coherente, ultrasonido, rayos x, entre otras. Se enumeran algunos de los principales escáneres activos sin contacto:

- **Tiempo de vuelo (***Time of flight***)**, este tipo de escáner estima la distancia entre el dispositivo y el objeto mediante la cronometración del viaje de ida y vuelta de un pulso de luz de longitud de onda conocido.

- **Triangulación**, este escáner utiliza un haz de luz láser y un detector de posicionamiento (DP), la luz es emitida y parte se refleja en el detector formando un triángulo entre la fuente de luz, el objeto y el detector de posicionamiento. La distancia es medida por la diferencia en el detector de posicionamiento ya que son conocidos los ángulos que se forman entre ellos.

- **Luz estructurada**, estos escáneres emiten un patrón de luz sobre el objeto y se analiza la deformación del patrón para determinar las medidas realizadas.

Entre los escáneres pasivos sin contacto se pueden mencionar los siguientes:

- **Estereoscópicos**, utilizan dos cámaras puestas en posiciones distintas mirando hacia el mismo objetivo y se evalúa la distancia entre los pixeles de cada imagen para estimar la distancia del objeto.

- **Por detección de silueta**, se utiliza una sucesión de fotografías alrededor de un objeto contra un fondo que resalte la silueta del objetivo, estas fotografías son utilizadas para estimar un objeto tridimensional.

Un sensor láser 2D o de perfiles permite generar una secuencia de mediciones en un mismo tiempo usando la técnica de triangulación, de esta forma se puede mejorar el tiempo de operación para construir un escáner 3D. En la figura 1.2 se ilustra de forma gráfica cómo un sensor láser 2D opera para obtener el perfil en una parte de un objeto.

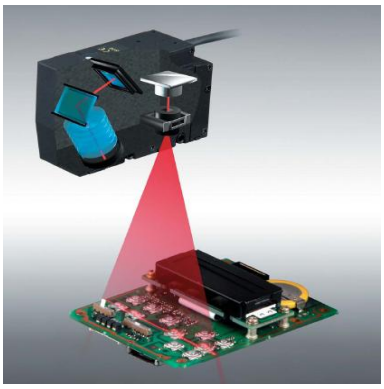


Figura 1.2: Ejemplo de escaneo utilizando un sensor láser 2D o de perfiles.

1.1.3 Nubes de puntos

Los archivos de nubes de puntos son un conjunto de vértices en un sistema de coordenadas tridimensionales. Estos vértices usualmente son definidos por las coordenadas xyz y son empleados para la descripción de la superficie exterior de un objeto [52, 5]. La mayoría de archivos de nubes de puntos son generados por escáners 3D. Estos dispositivos pueden generar de manera automática una gran cantidad de puntos de la superficie de un objeto en análisis y ponerlos en un archivo digital de datos. La figura 1.3 muestra la nube de puntos de un cubo escaneado desde arriba de una de sus caras.

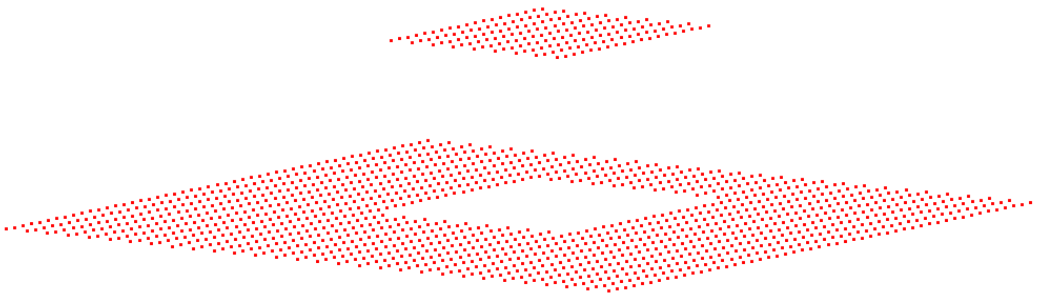


Figura 1.3: Ejemplo de una nube de puntos. Un cubo escaneado desde la parte superior, sólo se muestran los puntos de su cara superior y la base donde está puesto

1.1.4 Extracción y reconstrucción de objetos a partir de nubes de puntos

La reconstrucción de objetos requiere más trabajo que sólo obtener un archivo de nube de puntos. Un modelo completo del objeto bajo análisis requiere diferentes posiciones de escaneado, o diferentes vistas del objeto, para posteriormente buscar una o varias transformaciones geométricas que permitan poner a todas sus vistas en un mismo marco de referencia. Este problema de situar diferentes vistas en un mismo marco de referencia se le conoce como *registro* [43, 72]. Posteriormente, la reconstrucción implica la generación de modelos matemáticos que mejor describan la superficie del objeto o escena. Algunas técnicas útiles para este fin son la generación de mallas de polígonos, NURBS, splines y

CAD, entre otros.

1.2 Planteamiento del problema

En este trabajo se plantea utilizar un sensor láser 2D de marca comercial, una mesa de pruebas y dos sistemas de movimiento lineal de alta precisión para construir un escáner láser 3D. En la tabla 1.1 se muestra una comparación entre el escáner propuesto en este trabajo y los comercialmente disponibles en el mercado, que emplean la misma tecnología de medición por triangulación de un haz de luz láser. De la tabla 1.1 se puede notar que mediante el desarrollo de esta propuesta se puede disminuir el costo de la construcción de un escáner 3D, mejorar la precisión y proveer de un software abierto para futuras mejoras.

Tabla 1.1: Comparación entre el escáner 3D propuesto y algunas versiones comerciales. Los precios están dados en dolares estadounidenses

Escáner	**Precisión**	**Software adicional**	**Costo**
Propuesta	6 μm	—	3,000
Next Engine [3]	100 μm	3,000	3,000
Laser Design [2]	50 μm	costo extra	a partir de 15,000
Faro focus 3D [1]	2 mm	costo extra	20,000 a 30,000

El sensor láser que se propone usar en este trabajo de la marca OMRON [13], permite tomar 631 mediciones al mismo tiempo sobre una línea de 70 mm de longitud, mediante la técnica de triangulación. La forma de operación de este sensor nos permitirá hacer de forma más eficiente la adquisición de datos de la superficie de objetos del mundo real. Adicionalmente, el sensor es un dispositivo inteligente que se puede conectar a una computadora mediante el puerto RS-232, esto para recibir instrucciones de control y enviar datos de medición a la computadora. El sensor, de acuerdo a las hojas del fabricante [13], tiene un error estimado de $\pm 6\mu$m en la dirección de medición del eje z (considerando un sistema de coordenadas xyz). El sensor propuesto en este trabajo se seleccionó por tener un error promedio del orden de unidades de micrómetros y permitir múltiples mediciones simultáneas, en comparación de sensores con sólo un punto de medición.

Se propone construir una mesa de pruebas que permita absorber el ruido del medio ambiente, ya que el uso de este tipo de mesas son recomendables cuando los instrumentos de medición así lo requieren [57, 21], es decir, una perturbación de vibración prodría afectar el resultado de la prueba. La mesa, además de absorber el ruido y vibraciones del medio ambiente, puede ser utilizada como soporte para los componentes del escáner.

Las barras de desplazamiento lineal de alta precisión [33], son herramientas útiles para desarrollar sistemas donde se requiere un control preciso de movimiento. Mediante el uso de este tipo de sistemas de movimiento se puede construir un sistema cartesiano que permita el movimiento de un objeto en las coordenadas xy; y permitir conocer con precisión la posición del mismo.

Los datos generados por el escáner, pueden ser analizados para extraer características de los objetos, que permitan hacer una reconstrucción del mismo mediante el ajuste de parámetros de primitivas geométricas a los puntos.

En este trabajo se desarrolla el diseño e implementación de un sistema escáner 3D basado en un sensor láser 2D comercial de alta precisión, que mediante el movimiento del objeto en el plano cartesiano xy permite realizar la tarea de obtener los puntos de la superficie de objetos del mundo real y poder generar archivos de nubes de puntos que describen estos objetos. Posteriormente se describe la implementación de una metodología basada en el paradigma *Muestra Aleatoria de Consenso* (MAC, en inglés *Random Sample Consensus* (RANSAC)) [18] y un muestreo no uniforme que permite extraer primitivas geométricas de las nubes de puntos de una manera más eficiente que un muestreo uniforme y acelerar la convergencia del algoritmo MAC. Las principales primitivas geométricas que se trabajaron son: planos, esferas, cilindros y conos. Las primitivas que se proponen usar en el algoritmo usan técnicas directas de estimación de los parámetros de ecuaciones de primitivas geométricas basados principalmente en descomposición de matrices en valores y vectores propios y sistemas de ecuaciones lineales. También se presenta una metodología para el registro de nubes de puntos mediante descriptores de superficie y una combinación del algoritmo Puntos más Cercanos Iterativo (PMCI), (en inglés *Iterative Closest Point*

(ICP)) [43, 72] y un método de mínimos cuadrados como lo es el algoritmo propuesto por Levenberg-Marquardt [39], dando como resultado una metodología útil para resolver el problema del registro en 3D de forma más eficiente que utilizando solamente puntos.

1.3 Hipótesis

Dada la importancia de la digitalización de objetos del mundo real, en diferentes áreas de investigación y de la industria, se plantea como hipótesis lo siguiente:

Los objetos del mundo real pueden ser digitalizados, es decir, obtener la nube de puntos que describe su superficie, mediante el uso de escáneres láser. Los datos generados en el proceso de escaneo, pueden ser analizados mediante metodologías que permitan extraer algunas primitivas geométricas contenidas en su nube de puntos y empleando los modelos de estas primitivas, es posible reconstruir el objeto realizando el registro de las distintas partes del objeto ya escaneadas.

1.4 Objetivos

1.4.1 Objetivo general

Construir un escáner 3D empleando un sensor láser 2D de alta precisión en conjunto con un sistema de movimiento en un plano cartesiano para generar nubes de puntos y a partir de esos archivos de puntos hacer la extracción de primitivas geométricas que permitan reconstruir los objetos escaneados mediante modelos matemáticos.

1.4.2 Objetivos específicos

Para lograr el objetivo general se han de cumplir, entre otras etapas, con los siguientes objetivos particulares:

1. Desarrollar e implementar una mesa de pruebas con aislamiento de vibraciones del entorno.

2. Desarrollar un módulo de hardware que permita controlar el posicionamiento de un objeto en un plano cartesiano.

3. Desarrollar un módulo de software para interconectar el sensor láser con una computadora y sincronizar el movimiento de los objetos en un plano cartesiano para escanear el objeto y mediante este módulo de software generar archivos de nubes de puntos.

4. Extraer primitivas geométricas tales como: planos, esferas, cilindros y conos, contenidas en las nubes de puntos mediante el ajuste de los parámetros de sus ecuaciones matemáticas.

5. Desarrollar una metodología que permita hacer la integración de distintas nubes de puntos que describen un objeto.

1.5 Organización del documento

El contenido del presente documento está organizado en los siguientes capítulos:

- En el capítulo 2 se describe la construcción de una mesa de pruebas pasiva, la cual se utilizó para reducir el ruido y vibraciones del medio ambiente, además de servir como soporte para los componentes del escáner. Posteriormente se describen los módulos que se desarrollaron para ser utilizados en conjunto para la puesta en operación del

escáner. Al final de este capítulo, se muestran dos análisis que se desarrollaron para caracterizar el error del escáner durante el proceso de operación.

- En el capítulo 3 se describe el algoritmo MAC y la forma en que se le utilizó para ajustar los parámetros de ecuaciones de primitivas geométricas, a partir de una nube de puntos. En este capítulo se describe una reconstrucción de los objetos escaneados con modelos computacionales. También se presenta una mejora al algoritmo MAC, considerando que los datos que se obtuvieron con el escáner desarrollado están ordenados, es decir, que cada punto tiene sus vecinos más cercanos en una matriz de datos. Esto para lograr disminuir el número de iteraciones necesarias en el proceso del algoritmo MAC.

- En el capítulo 4 se trata con el problema del registro de nubes de puntos y el uso de descriptores de superficie que nos permitieron resolver este problema de forma más eficiente que usando sólo los datos de los vértices de los puntos. Finalmente, se muestran las pruebas y los resultados al aplicar esta metodología al registro de nubes de puntos obtenidas por el escáner.

- Finalmente en el capítulo 5 se presentan las conclusiones de este documento y los trabajos futuros.

2

Diseño y construcción de un escáner 3D para digitalizar objetos

En este capítulo se describe la construcción de un sistema modular que permite hacer la tarea de la digitalización de objetos del mundo real. El sistema en su diseño consta de cuatro módulos que les llamamos: módulo mecánico, de hardware, del sensor láser y de software. Además consta de una mesa pasiva de pruebas. Comenzaremos el capítulo describiendo la mesa pasiva de pruebas, posteriormente se describirán estos cuatro módulos y finalmente la caracterización del error obtenida al poner en operación el sistema de escaneo.

2.1 Mesa de pruebas ultra estable

El uso de mesas de prueba ultra estables son recomendables para la experimentación en óptica, metrología y otras aplicaciones donde la absorción de ruido y vibraciones es necesaria. Nuevos y complejos mecanismos son propuestos en la industria donde existen

ambientes adversos de ruido y vibraciones y se requiere resolver este tipo de problemas. Dos de los principales problemas a resolverse en el diseño de mesas de prueba ultra estables son: el aislamiento de las vibraciones provenientes del ambiente y el tipo de material empleado sobre los componentes de aislamiento de ruido y vibraciones. El ruido ambiental y las vibraciones son principalmente transmitidas por las áreas de contacto de la mesa de prueba con la tierra, para resolver las vibraciones provenientes del ambiente se emplean soportes pasivos o activos, la principal característica en los soportes activos es el uso de mecanismos de control que aíslan la mesa de prueba con el ambiente y que permiten corregir los niveles de altura de la mesa empleando cámaras de hule llenas de gas. Un inconveniente con esté tipo de soportes radica en que las cámaras de hule se degradan con el tiempo y necesitan mantenimiento especializado continuamente, además de requerir de condiciones especiales para su funcionamiento. Los soportes pasivos no emplean mecanismos de control, lo cual sería la principal diferencia con los soportes activos.

En este trabajo se optó por el diseño y construcción de una mesa de prueba con soportes pasivos, que es más sencillo y menos costoso que los soportes activos. Para resolver el problema del material a emplear en el diseño de la mesa de prueba es necesario que se cumplan los siguientes criterios [21]:

1. Rigidez contra deformaciones
2. Absorción de las frecuencias de resonancia
3. Amortiguamiento crítico contra vibraciones
4. El acoplamiento con el resto de la estructura

El primer criterio se cumple empleando materiales duros y densos. En la mesa diseñada se empleó una placa rectangular de acero con un grosor de 5 mm y medidas de 70×63 cm. Para evitar la frecuencia de resonancia longitudinal y de torsión se utilizó un contenedor rectangular metálico con las siguientes dimensiones: 70 cm de ancho, 63 cm de largo y

10 cm de alto. El contenedor está dividido internamente en cuatro partes con dos placas metálicas de un grosor de 5 mm y altura de 3 cm. La figura 2.1 muestra las dimensiones del contenedor empleado y su división interna. Esta división interna del contenedor permite evitar las deformaciones en la mesa de prueba. El diseño, implementación y evaluación de mesas similares se pueden encontrar en la referencia [21]. Las vibraciones en los laboratorios y lugares de trabajo comúnmente se categorizan en frecuencias bajas, medias y altas; las vibraciones bajas están alrededor de 4 y 400 Hz y las frecuencias medias y altas están entre 400 Hz y 15 KHz. Sin embargo, las frecuencias que mayor impacto tienen en los sistemas mecánicos son las que están entre los 200 y 2 KHz [57]. Para aislar nuestra mesa de prueba de las vibraciones externas fueron puestas diferentes capas de materiales en el contenedor. En la figura 2.2 se muestran los diferentes materiales empleados: una capa de arena fina (8 cm) y dos capas de hule espuma compacta (3.5 cm cada una) separadas por una capa de madera de 2.5 cm. Con la implementación de esta mesa de prueba se puede cumplir con los cuatro requerimientos básicos de diseño. La base metálica además de cumplir con el criterio de rigidez contra deformaciones nos sirve como soporte mecánico para los demás componente del sistema del escáner que se le añadirán.

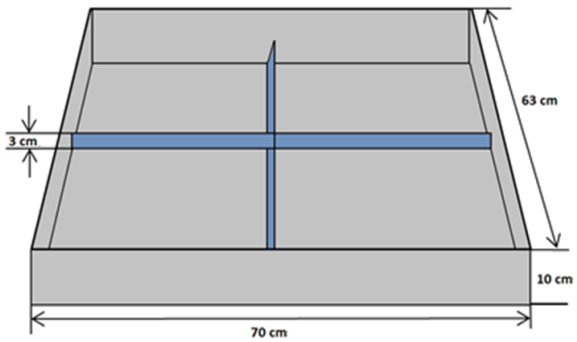


Figura 2.1: Dimensiones del contenedor empleado para absorber las vibraciones y ruido

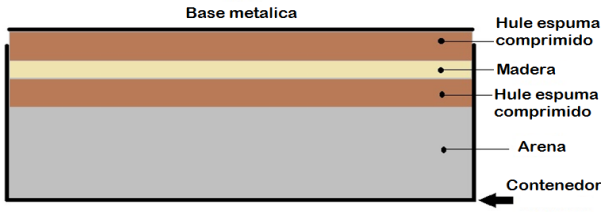


Figura 2.2: Capas de materiales empleados en la construcción de la mesa de prueba pasiva para amortiguar el ruido y vibraciones del medio ambiente

2.2 Módulos empleados para construir el sistema de escaneo 3D

En esta sección se describen los principales módulos del escáner 3D: mecánico, hardware, sensor 2D y software de operación.

2.2.1 Módulo mecánico

En el diseño de sistema de escaneo es necesario mover el objeto hacia diferentes direcciones para lograr una digitalización completa. Para esto se emplearon dos módulos lineales de alta precisión con rieles y patines basados en un tornillo sin fin con movimiento por balines y bloqueo de alta precisión [33]. En la figura 2.3 se muestra el diseño de este tipo de sistemas lineales. La resolución de cada módulo lineal es de 10 mm de movimiento lineal por cada vuelta completa del tornillo. En la figura 2.4 se muestra la forma del módulo lineal empleado. Para obtener un sistema de movimiento con dos grados de libertad se configuraron los dos módulos lineales de forma perpendicular entre ellos, para poder obtener posiciones en coordenadas cartesianas xy.

Cada módulo lineal tiene unido un motor a pasos bipolar con una resolución de 200 pasos por vuelta. Esto nos permite obtener una relación cinemática de 50 micrómetros en movimiento lineal por cada paso del motor. La figura 2.5 muestra las configuración empleada para lograr el sistema de coordenadas ortogonal.

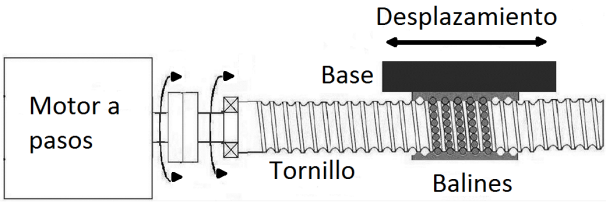


Figura 2.3: Diseño de los módulos lineales con tornillo sin fin y balines

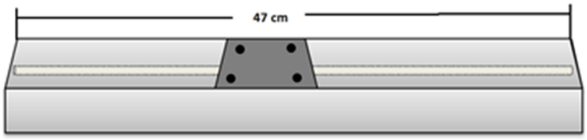


Figura 2.4: Representación gráfica de los módulos lineales empleados

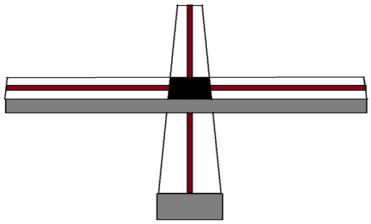


Figura 2.5: Configuración de los dos módulos lineales empleados para obtener un sistema de movimiento cartesiano

2.2.2 Módulo de hardware

Para lograr el movimiento de los módulos lineales es necesario construir un módulo de hardware que haga la interfaz entre los motores a pasos y las señales de control de movimiento. Los módulos que permiten la operación de motores a pasos que se venden comercialmente, comúnmente son sistemas cerrados que no pueden ser modificados y que funcionan para un determinado tipo de motor, además de requerir de un software de operación que puede tener un costo adicional con la desventaja de no permitir una configuración particular al usuario. Para resolver este problema se decidió diseñar y construir un módulo de hardware que sea programable, escalable y fácil de implementar basado en un micro-controlador PIC-18F4550 [59]. Este dispositivo tiene cuatro puertos de entrada y salida configurables, 32 KB de memoria programable y un puerto USART (*Universal*

Synchronous Asynchronous Receiver Transmitter, por su nombre en inglés), entre otras características. El micro-controlador fue programado para que en uno de sus puertos se genere la secuencia de pulsos enviados al módulo de hardware, descrito a continuación, para lograr el movimiento en sentido las manecillas del reloj y viceversa. Las señales de control son enviadas a una unidad de potencia basada en un puente completo L298 [58], con valores de operación nominales de 4 Amps y 46 Volts en corriente directa. Los motores bipolares empleados requieren un máximo de 5 Volts y 3 Amps para operar, por lo cual la selección del puente L298 es suficiente para la etapa de potencia del movimiento del sistema.

Para obtener un módulo de hardware que fuese capaz de intercomunicarse con una computadora, se habilitó el puerto de comunicaciones USART para operar como una interfaz RS-232 bi-direccional no simultánea. Esta comunicación nos permite sincronizar el movimiento de los módulos lineales con el sensor láser empleado y la escritura de los archivos de datos.

Tabla 2.1: Comandos que pueden ser enviados al controlador de hardware del escáner

Comando	**Valor**
Avanzar eje x	1
Retroceder eje x	2
Avanzar eje y	3
Retroceder eje y	4

En la figura 2.6 se muestra el diseño del hardware desarrollado para el control de los motores a pasos. El hardware tiene un conector DB9 para su interconexión con una computadora. Este conector recibe las señales de control y son traducidas por un circuito integrado MAX-232 a señales que pueden ser procesadas por el micro-controlador. Cuando el puerto serial en el micro-controlador tiene datos, éstos son leídos por la unidad de procesamiento y se revisa cuál es el comando recibido y se ejecuta para generar la secuencia de pulsos para movilizar los motores a pasos. Después que termina de realizar la tarea devuelve un valor de "0" para indicar a la computadora que terminó la tarea. En la

tabla 2.1 se muestran los comandos empleados para el control de los motores a pasos.

2.2.3 Módulo de sensor láser

Un sistema láser (de las iniciales en inglés: *Light Amplification by Stimulated Emission of Radiation*, amplificación de luz por emisión estimulada de radiación) es un dispositivo que emite radiación electromagnética [26] o luz coherente. El sensor empleado en este trabajo está compuesto por una fuente de luz coherente color rojo de 670 nm de longitud de onda y un detector de posición (DP). La unión de estos dos componentes permiten construir un sistema de medición de datos de rango mediante el método de triangulación. La figura 2.7 presenta la configuración básica de este método de medición: un haz de luz es proyectado sobre un objetivo M y un DP uni-dimensional captura su ubicación. La distancia horizontal $B = OC$ entre la luz coherente y el centro de la lente focal y el ángulo interior α que se forma entre la luz coherente y el eje óptico del lente focal son conocidos; la distancia vertical $L + \Delta z$ entre el centro del lente focal C y el objetivo M puede ser evaluado en la posición de la luz en el DP con la posición de Δu.

Una de las ventajas del sensor empleado en este trabajo es la emisión de multiples rayos de luz coherente que nos permite simultáneamente capturar 631 mediciones de rango en un mismo instante. En la figura 2.8(a) se ejemplifica la obtención de datos de rango simultáneamente de un objeto. La figura 2.8(b) ejemplifica los puntos obtenidos sobre la superficie del objeto. Esto se puede entender como el perfil del objeto en una posición cualquiera. Si realizamos un barrido de la superficie del objeto y en cada posición obtenemos los datos de rango del perfil se puede generar una vista tridimensional del objeto con nubes de puntos. Para lograr que el sensor sea sincronizado con el movimiento de los motores y el almacenamiento de datos, se le activó la función de interface RS-232 en el sensor, lo que nos permite enviar comandos al sensor mediante una computadora, para realizar mediciones y leer los datos del sensor y posteriormente almacenar la información en archivos de nubes de puntos.

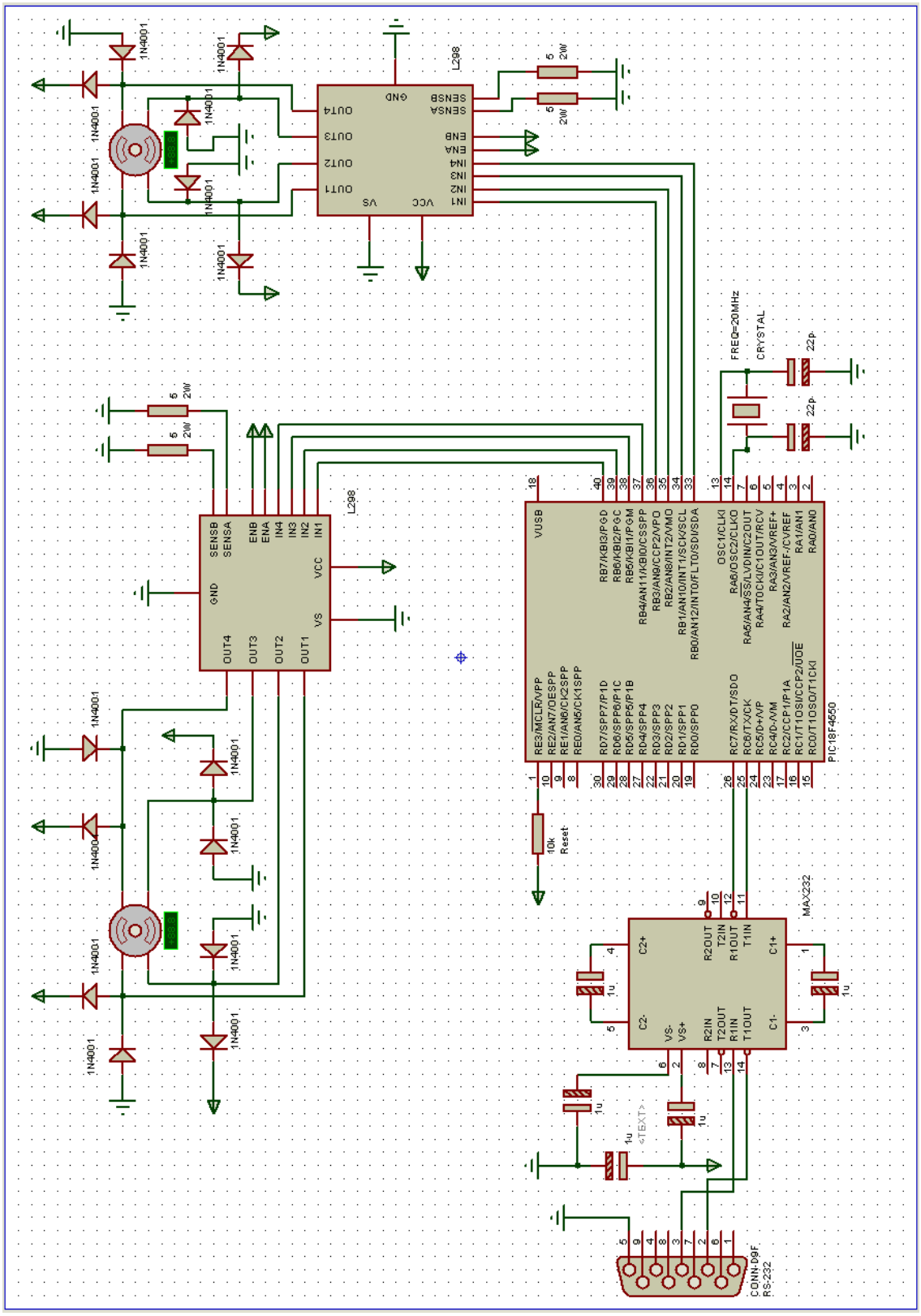


Figura 2.6: Diseño esquemático del circuito eléctrico para controlar el escáner

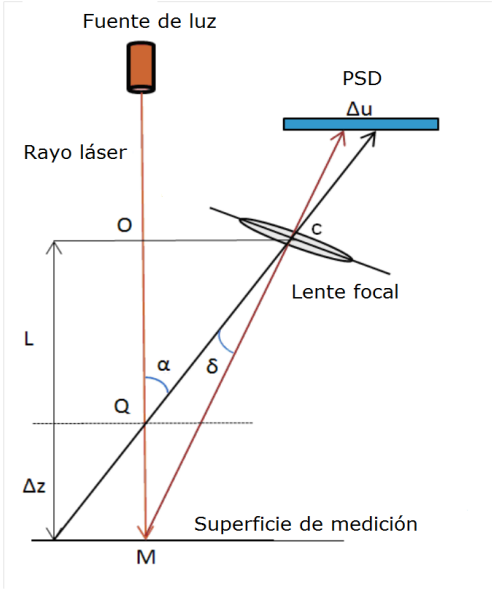


Figura 2.7: Ilustración gráfica del método de triangulación para obtener datos de rango empleando un láser y un DP

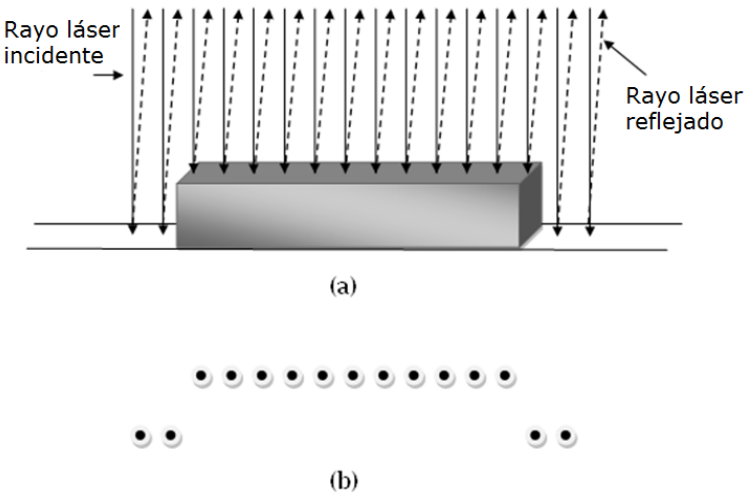


Figura 2.8: (a) Ejemplificación de la múltiple obtención de datos de rango con el sensor láser empleado. (b) Puntos obtenidos de la superficie del objeto

El sensor empleado en este trabajo es un sensor láser 2D modelo WDS70/ZG-Series [13] fabricado por la empresa OMRON, este es un dispositivo inteligente empleado en la industria principalmente para automatizar tareas de control de calidad entre otras aplicaciones. En la figura 2.9 se muestra la configuración del sistema de escaneo. De acuerdo con los

datos técnicos del fabricante, el sensor de 2D, trabaja mejor a una distancia de 210 mm de distancia entre el sensor y el objetivo. Esto permite generar un segmento de luz coherente de un ancho de 70 mm con una precisión de 6 μm en dirección vertical.

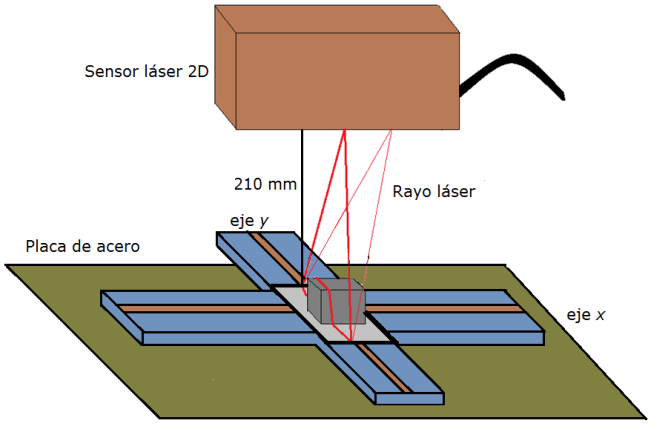


Figura 2.9: Puesta a punto del sensor láser en la mesa de pruebas

2.2.4 Módulo de software

Para poder operar el sistema de movimiento y el sensor láser es necesario desarrollar un módulo de software que permita la sincronización de operaciones entre los módulos anteriores, y generar los archivos de nubes de puntos. Para lograr esto, se implementó un software en lenguaje Phyton. Se eligió este lenguaje debido a que permite una mayor portabilidad entre sistemas operativos a diferencia de otros lenguajes incluso como Java o C/C++. El módulo de comunicación se logra por medio de los puertos RS-232 de la computadora los cuales son fácilmente configurados y utilizados en este lenguaje.

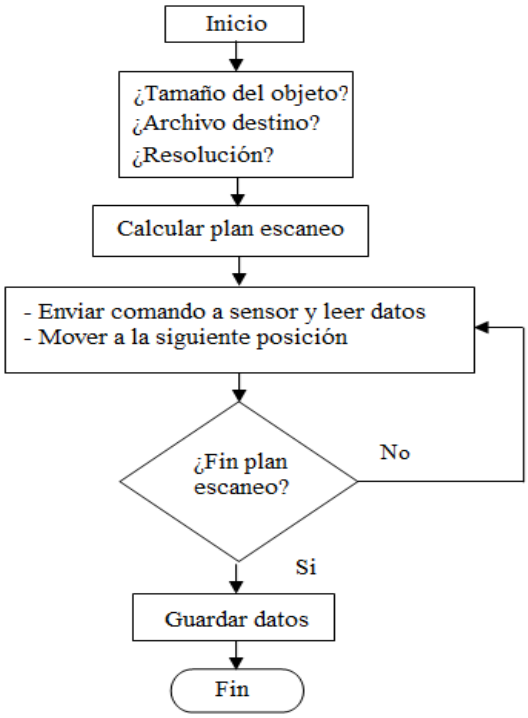


Figura 2.10: Flujo de trabajo del software desarrollado en Phyton para operar el escáner 3D

La operación del software comienza preguntando al usuario por las dimensiones del objeto a escanear, la resolución y el nombre del archivo destino donde se guardarán los puntos. La salida es un archivo de texto plano donde cada línea de información está compuesta de 3 datos que son los valores de las coordenadas en x, y y z respectivamente. Estos valores están dados en punto flotante con 8 posiciones decimales. La resolución que se ajusta en el sistema está referida a la cantidad de muestras de perfil que se tomarán por milímetro. El usuario puede decidir tomar valores enteros entre 1 y 5, 1 es la máxima resolución y 5 es la mínima resolución; $v.g.$ 1 $=$ muestras espaciadas 0.2 milímetros y 5 $=$ muestras espaciadas 1 milímetro. La resolución puede ser importante en casos donde el usuario quiere evitar la generación de gran cantidad de datos y archivos muy grandes y difíciles de manejar. Las dimensiones del objeto son dadas al sistema por un área rectangular (en unidades de centímetros) que abarque completamente al objeto. La puesta a punto del sistema de escaneo nos permite fácilmente remover puntos que no son del

objeto debido a que datos fuera del rango de 210 mm son valores negativos que se omiten al ser escritos al archivo destino. Después de la especificación de los pasos anteriores se genera un plan de escaneo por cada eje. Si la dimensión del objeto en la coordenada y es más pequeña que 70 mm entonces sólo es calculada la cantidad de perfiles a medir en el eje x empleando la formula $perfiles$=(dimensión en x)(0.2(resolución)). La constante de 0.2 es la separación que hay entre cada muestra de medición, es decir, 0.2 milímetros para una resolucion=1 y 1 milímetro para una resolucion=5. Si el valor de la dimensión en el eje y es mayor que 70 mm entonces se calcula un plan de escaneo para este eje. El módulo lineal en y se desplazará 35 mm hasta alcanzar a cubrir el total del área del objeto. El motivo de usar este tipo de escaneo en el eje y es para permitir aplicar algoritmos de registro en 3D como el algoritmo Puntos más Cercanos Iterativo (PMCI) [43, 72] que requieren que haya un buen traslape entre vistas para poder ser registradas. En la figura 2.10 se ilustra la operación del software desarrollado. En la figura 2.11 se presenta el diseño físico del escáner desarrollado.

Figura 2.11: Configuración final del diseño físico del escáner 3D

2.3 Caracterización del escáner 3D

En esta sección se describe un análisis numérico que se realizó para caracterizar el error del escáner, mediante el ajuste de los datos obtenidos del escaneo de dos modelos del mundo real a dos modelos geométricos. Esto se hizo con el propósito de obtener valores de referencia en el error obtenido al operar el sistema. El fabricante del sensor láser especifica que el dispositivo tiene un error de ± 6 μm en dirección vertical.

El sensor de medición láser al ser un dispositivo basado en un método óptico tiene la desventaja de no poder realizar correctamente las mediciones en cualquier objeto con superficie brillante y condiciones de iluminación altas, y en algunas ocasiones el sensor no obtiene los 631 puntos del perfil de medición. Para resolver este problema se empleó un método de aproximación por extrapolación lineal de datos utilizando los 20 datos más cercanos, lo cual permite hacer una aproximación de datos faltantes.

Los dos métodos empleados para evaluar el error en la medición del sensor están hechos con un método lineal mediante una eigendescomposición y uno no lineal mediante una optimización no lineal por mínimos cuadrados.

2.3.1 Caracterización con un método lineal

Para realizar el primer análisis de datos, se realizó una regresión lineal de datos a la ecuación de un plano que minimiza la distancia ortogonal de cada punto a ese plano.

La descripción del problema es la siguiente: dado un conjunto de puntos 3D en coordenadas (x, y, z), es necesario encontrar la mejor aproximación de la ecuación de un plano tal que la suma del cuadrado de las distancias de todos los puntos a tal plano sea mínima.

El problema puede ser representado gráficamente en la figura 2.12, donde los puntos en color blanco son datos observados en un experimento y los puntos en color negro representan los valores de los puntos en el plano que mejor se ajusto a los datos observados.

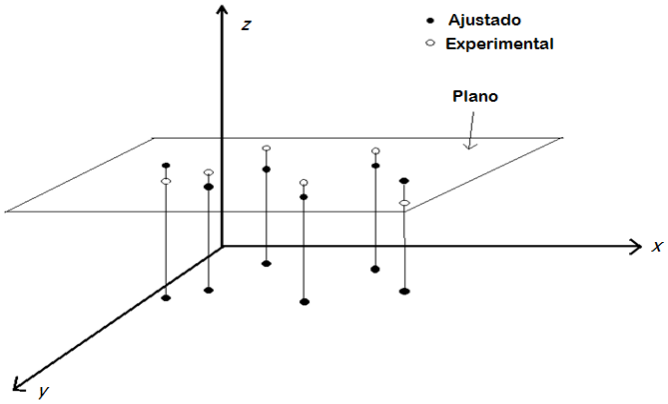


Figura 2.12: Regresión o ajuste de puntos 3D a un plano

El primer análisis se centra en la idea de los trabajos publicados en [7, 48], en los cuales se busca desarrollar mediante métodos directos por inversión o descomposición de matrices el ajuste de datos a ecuaciones de cónicas. Para comenzar podemos definir la ecuación del plano mediante la expresión:

$$F(\mathbf{a};\mathbf{x}) = \mathbf{a}\cdot\mathbf{x} = ax + by + cz + d = 0, \tag{2.1}$$

donde $\mathbf{a} = [a, b, c, d]^\mathsf{T}$ y $\mathbf{x} = [x, y, z, 1]^\mathsf{T}$. $F(\mathbf{a};\mathbf{x}_i)$ se conoce como la *distancia algebraica* del punto $\mathbf{x}_i$ al plano representado por $F(\mathbf{a},\mathbf{x}) = 0$. El problema lo podemos definir ahora como la minimización de la suma del cuadrado de las distancias algebraicas:

$$\|D\mathbf{a}\|^2 = \mathbf{a}^\mathsf{T} D^\mathsf{T} D\mathbf{a} = \sum_{i=1}^{N} F(\mathbf{a};\mathbf{x}_i)^2, \tag{2.2}$$

donde $D = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$, para N puntos, sujetos a la restricción $a^2 + b^2 + c^2 = 1$. Esta restricción viene dada por el hecho que el vector $\mathbf{n} = [a, b, c]^\mathsf{T}$ es el vector normal que define el plano que mejor ajusta los datos, y cuando su vector normal es igual a la unidad, entonces el valor $|\mathbf{n}^\mathsf{T}[x_i, y_i, z_i]^\mathsf{T} + d|$ es igual a la distancia real del punto $\mathbf{x}_i$ al plano representado por $\mathbf{n}$ y d.

Introduciendo multiplicadores de Lagrange λ y diferenciando la ecuación (2.2) se llega al sistema de ecuaciones simultáneas:

$$\begin{aligned} 2D^T D\mathbf{a} - 2\lambda C\mathbf{a} = 0, \\ \mathbf{a}^T C\mathbf{a} = 1, \end{aligned} \tag{2.3}$$

donde C es la matriz de restricción

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

y podemos reescribir el problema como:

$$S\mathbf{a} = \lambda C\mathbf{a}, \text{ and } \mathbf{a}^T C\mathbf{a} = 1, \tag{2.4}$$

donde S es la matriz de dispersión $D^T D$. Este sistema es fácilmente resuelto considerando el problema generalizado de eigenvectores de la ecuación (2.4). Pero debido a la estructura de la matriz C este problema es equivalente a resolver la eigendecomposición del problema definido por:

$$\tilde{S}\tilde{\mathbf{a}} = \lambda\tilde{\mathbf{a}}, \tag{2.5}$$

donde $\tilde{\mathbf{a}} = \mathbf{n}$ y $\tilde{S}$ es la matriz superior de tamaño 3×3 de S. Por lo tanto, la solución es el eigenvector asociado al minímo eigenvalor de la ecuación (2.5). Ahora sólo para determinar el parámetro d de la ecuación del plano, se calcula el centroide de los puntos dados (x_0, y_0, z_0). Dado que es un punto que pertenece al plano y haciendo la evaluación algebraica podemos obtener $d = -(ax_0 + by_0 + cz_0)$. Para evaluar el método descrito en esta sección, los datos del escáner fueron normalizados a media cero.

Para hacer la evaluación de los datos y del método descrito se realizó una experimentación con un objeto de superficie plana y se realizó el escaneo con el sistema láser. Se tomaron un total de 63,100 puntos y se guardaron en un archivo de nubes de puntos. A

los datos observados se les aplicó el método de regresión lineal explicado, y se tomó el mínimo eigenvector asociado al mínimo eigenvalor de la solución obtenida y se calculó el valor de d de la forma explicada.

Después de hacer la regresión de datos se hizo la medición del error de cada punto observado hacia el plano ajustado. En la tabla 2.2 se presenta un resumen del error estimado: media, máximo, mínimo, la raíz del promedio del cuadrado de los errores (RMSE) y la desviación estándar. En la figura 2.13 se muestra solamente un perfil tomado del escáner y el error absoluto obtenido contra el modelo del plano.

Tabla 2.2: Error estimado empleando la técnica descrita es esta sección y 63,100 puntos generados por el escáner

Promedio	**Máximo**	**Mínimo**	**RMSE**	**Desviación estándar (σ)**
3.67μm	25.61 μm	0 μm	4.21 μm	2.48 μm

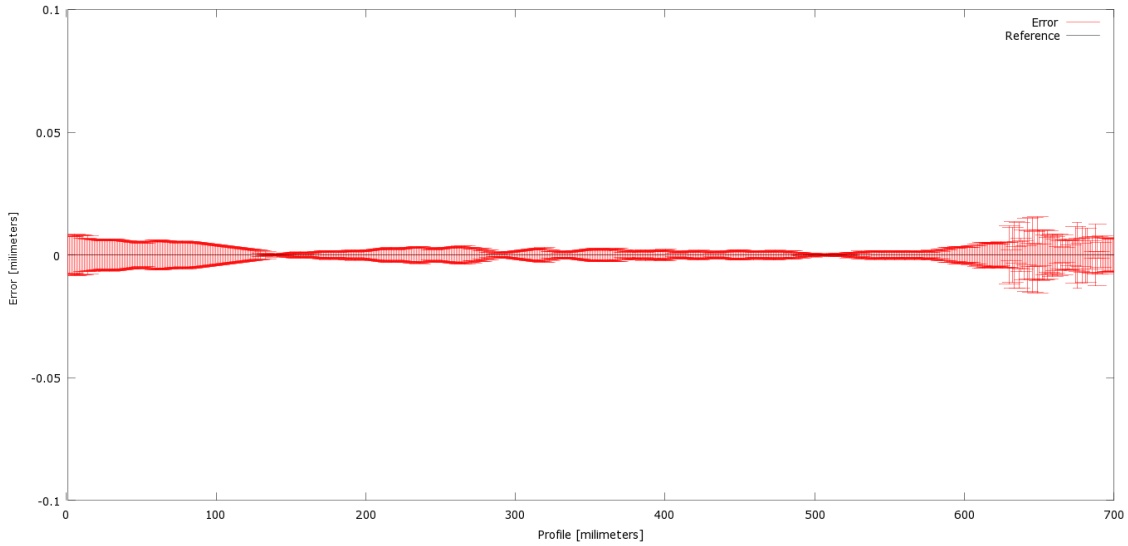
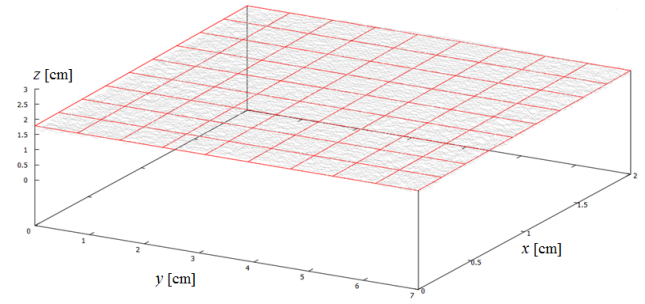


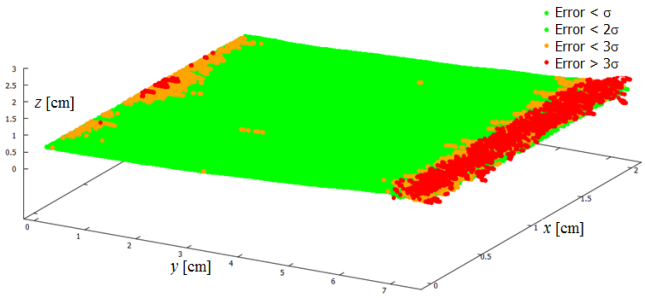
Figura 2.13: Errores absolutos medidos en uno de los perfiles del escáner en comparación con el ajuste del modelo del plano descrito en esta sección

En general podemos notar en la figura 2.13 que el mayor error del sensor está en los bordes del perfil medido. Esto nos puede ayudar a considerar emplear el centro del perfil para mejores resultados.

En la figura 2.14.(a) se muestra la gráfica del modelo del plano ajustado a los datos del escáner empleando el método de regresión lineal descrito. La figura 2.14.(b) muestra los puntos que están dentro del valor de σ y 2σ de color verde, en color anaranjado los puntos que tienen una distancia mayor que 2σ e igual o menor que 3σ y de color rojo los puntos con una distancia al plano mayor de 3σ.



(a)



(b)

Figura 2.14: (a) Nube de puntos y el modelo del plano ajustado con el método de regresión descrito. (b) Gráfica del error por áreas, en color verde se muestran los puntos con una distancia menor a σ y 2σ, en color anaranjado los puntos con una distancia al plano mayores que 2σ e igual o menor que 3σ, en color rojo los puntos con una distancia mayor de 3σ

Los resultados obtenidos nos muestran un error promedio de 3.67 μm y una desviación estándar de 2.48 μm, a pesar de que el máximo error es de 25.61 μm. Esta medición se encuentra situada en los bordes de la línea de escaneo, lo cual concuerda con las hojas técnicas del fabricante quien menciona que la máxima precisión está en el centro de la línea de escaneo. Cabe mencionar que el conteo de los puntos que están por arriba de 3σ sólo representan el 3.12 % del total de puntos medidos en este experimento.

2.3.2 Caracterización con un método de regresión mediante un ajuste no lineal

El segundo método de análisis numérico para evaluar la precisión del escáner está basado en una minimización no lineal de la suma de los errores cuadráticos: se considera la obtención inicial de los parámetros de una esfera ($v.g.$ su radio y su centro) mediante un modelo lineal y posteriormente la optimización de estos valores mediante un algoritmo no lineal de optimización de mínimos cuadrados. Aquí se describirá el método desarrollado en [6]. Para describir este método se considera un conjunto de puntos sobre la superficie de una esfera en coordenadas tridimensionales (ver figura 2.15). El punto $\mathbf{o}$ es el origen del sistema de coordenadas, $\mathbf{c} = [x_c, y_c, z_c]^\mathsf{T}$ es el centro de la esfera, r es su radio y $\mathbf{x} = [x, y, z]^\mathsf{T}$ es un punto arbitrario sobre la superficie de la esfera. La ecuación vectorial de la esfera se puede reescribir mediante la expresión:

$$(\mathbf{x} - \mathbf{c})^T \cdot (\mathbf{x} - \mathbf{c}) = r^2 \tag{2.6}$$

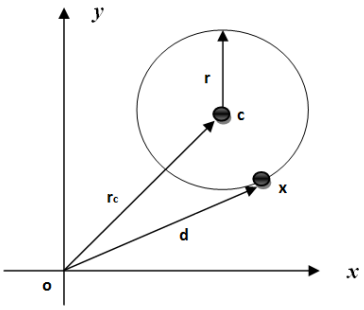


Figura 2.15: Representación de los parámetros de la esfera

La ecuación (2.6) tiene cuatro variables desconocidas: los elementos del vector $\mathbf{c}$ y r y se requieren al menos cuatro puntos para poder calcularlos. Para encontrar los valores de $\mathbf{c}$ se construye un sistema lineal de ecuaciones:

$$A\mathbf{c} = \mathbf{b} \tag{2.7}$$

donde:

$$A = \begin{bmatrix} \mathbf{x}_1 - \mathbf{x}_2{}^{\mathsf{T}} \\ \mathbf{x}_1 - \mathbf{x}_3{}^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_1 - \mathbf{x}_n \end{bmatrix}, \qquad \mathbf{b} = \frac{1}{2} \begin{bmatrix} d_1 - d_2 \\ d_1 - d_3 \\ \vdots \\ d_1 - d_n \end{bmatrix}. \tag{2.8}$$

donde $d_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$. Si tenemos n puntos, con índices $(i = 1, 2, 3, \ldots, n)$ se puede cambiar el índice con el que se calcula la ecuación (2.8) y se obtienen n valores para el vector $\mathbf{c}$ y podríamos usar todos ellos para hacer un promedio del valor de las coordenadas del centro de la esfera mediante la ecuación:

$$\mathbf{c} = \frac{\sum_{i=1}^{n} \mathbf{c_i}}{n} \tag{2.9}$$

Posteriormente, el radio se puede obtener empleando la ecuación (2.6) sustituyendo cada punto $\mathbf{x}_k$, para $k = 1, 2, 3, \ldots, n$, individualmente para generar n valores para r y finalmente el radio aproximado se calcula mediante la ecuación:

$$r = \frac{\sum_{i=1}^{n} r_k}{n} \tag{2.10}$$

Los valores obtenidos ahora se pueden emplear para ser optimizados mediante un algoritmo de ajuste no lineal. El problema ahora lo podemos definir mediante la función de error:

$$F(\mathbf{c}, r) = \sum_{i=1}^{n} \left(\sqrt{(\mathbf{x}_i - \mathbf{c})^T \cdot (\mathbf{x}_i - \mathbf{c})} - r \right)^2 \tag{2.11}$$

Un segundo ejercicio se realizó con el sistema láser: se escaneó un objeto de superficie esférica y se obtuvieron un total de 63,742 puntos. Se aplicó el método descrito anteriormente y por cada punto se resolvió la ecuación lineal (2.6) para obtener los valores de $\mathbf{c}$ y r mediante las ecuaciones (2.9) y (2.10) respectivamente. Después, se utilizó la imple-

mentación en Matlab de la función $lsqnolin$ la cual resuelve problema de optimización no lineal por mínimos cuadrados, la ejecución de la optimización sólo toma 4 iteraciones en minimizar el problema con un tolerancia de error de 10^{-8}.

Después de optimizar los parámetros de la esfera y construir su modelo final, se realizó un ejercicio para medir cuál era el error de cada punto observado hacia la superficie de la esfera construida. La tabla 2.3 muestra las estadísticas del error medido de los datos reales con los datos del modelo.

Tabla 2.3: Error estimado empleando la técnica descrita en esta sección y 63,742 puntos generados por el escáner

Promedio	**Máximo**	**Mínimo**	**RMSE**	**Desviación estándar (σ)**
1.23 μm	10.8 μm	0 μm	1.82 μm	0.24 μm

La figura 2.16 muestra la gráfica del modelo de la esfera con una malla de color rojo y los puntos de color verde son los datos obtenidos mediante el sistema de escaneo construido.

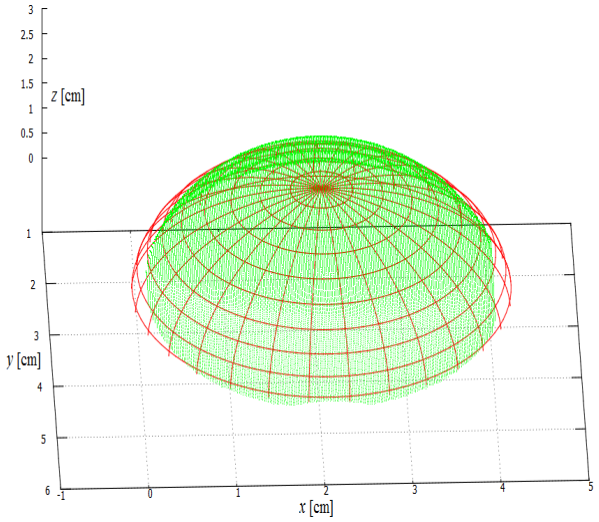


Figura 2.16: La malla cuadricular representa el modelo de la esfera ajustada a los datos observados mediante el sistema de escáner construido. La nube de puntos se ajusta alrededor del modelo

3

Extracción de primitivas geométricas a partir de nubes de puntos

En la digitalización de objetos mediante escáneres láser se producen archivos de nubes de puntos, los cuales describen la forma de la superficie del objeto escaneado en un sistema de coordenadas tridimensionales. Estos puntos 3D pueden ser utilizados en diferentes áreas de la industria, por ejemplo en ingeniería inversa, CAD, control de calidad, metrología y video juegos, entre otras; y en áreas de la investigación como: visión por computadora, reconocimiento de patrones y robótica, entre otras [54, 70, 43].

Pero un problema que se tiene con los archivos de nubes de puntos generados por dispositivos de escaneo láser, es la densidad de información obtenida, lo cual produce archivos de miles o incluso millones de puntos. Esto lleva a problemas como el análisis eficiente de los datos, problemas de procesamiento en dispositivos con pocos recursos de cómputo como un robot [31], la carga a la memoria principal de una computadora, exceso de tiempo de procesamiento, además de ralentizar el proceso de graficación. Para disminuir estos inconvenientes con los archivos de puntos densos se busca desarrollar metodologías

que nos permitan mejorar el análisis de las nubes de puntos y la interacción con el usuario. Una técnica comúnmente empleada en la visión por computadora es la segmentación. El proceso de segmentación permite la partición de una imagen o escena 3D en multiples partes o segmentos $i.e.$ un conjunto de puntos o píxeles en una región que son similares de acuerdo a ciertos criterios como homogeneidad, color, textura, bordes, etc. [50]. Para realizar el proceso de segmentación en las nubes de puntos se podrían definir dos procesos: uno de manera manual y otro de forma automática. En un caso de estudio publicado en [35] los autores desarrollaron la segmentación de múltiples primitivas geométricas para un proceso de ingeniería inversa, el usuario debe seleccionar manualmente partes de la nube de puntos para extraer cada primitiva y posteriormente se aplica un ajuste de puntos a un modelo. El mismo procedimiento lo deben hacer para cada primitiva del objeto analizado y así obtener su representación completa. Este trabajo manual puede volverse inapropiado debido a la múltiple interacción con los datos, además de que si el objeto tiene un gran número de primitivas a extraer y si el tamaño de cada primitiva tiene poca densidad de puntos la tarea podría volverse inviable para una persona. La segmentación automática puede realizarse mediante técnicas y métodos de procesamiento de imágenes y visión por computadora [29] para extraer formas de interés pare el usuario. Este procesamiento puede ser más preciso y eficiente que un método manual.

En este capítulo se presenta una metodología para la extracción de primitivas geométricas como: planos, esferas, cilindros y conos mediante un proceso de segmentación automática empleando el paradigma del algoritmo Muestra Aleatoria de Consenso (MAC) (*Random Sample Consensus* (RANSAC) en inglés) [18]. También se propone un modelo por cada primitiva para ser almacenadas de forma más eficiente que el de la nube de puntos y como un paso adicional para la reconstrucción del objeto empleando técnicas de modelado con Geometría Constructiva Sólida (CSG).

3.1 Trabajos previos

La segmentación de primitivas geométricas es un problema común en ciencias de la computación. Durante años anteriores se han hecho algunos trabajos sobre este tema. En esta sección sólo se mencionan algunas de las publicaciones relevantes a este trabajo.

3.1.1 Visión por computadora

En visión por computadora las metodologías más conocidas y empleadas para la segmentación de primitivas son el paradigma MAC [18] y la transformada de Hough [14]. Ambas metodologías han mostrado buen desempeño para detectar primitivas en 2D y 3D, aún en presencia de gran cantidad de datos atípicos. Sin embargo una de las principales desventajas de la transformada de Hough es su gran demanda de recursos de cómputo [40].

Un trabajo relevante empleando la transformada Hough para detectar únicamente planos es presentado en [23]. Una desventaja de esta propuesta es que hace la estimación de los vectores normales de cada punto de la nube de puntos. Los autores también proponen como trabajo futuro una técnica para extraer cilindros y esferas.

Otra metodología robusta muy utilizada en visión por computadora es el *framework tensor voting* [20], el cual es útil para reconstruir escenas 3D, pero la principal desventaja de esta técnica es que solo ajusta curvas libres y no ecuaciones de primitivas geométricas.

3.1.2 Graficación por computadora

En el área de graficación por computadora, en los trabajos publicados en [9, 69] se propone una metodología para reconstruir superficies empleando planos; los autores afirman poder extraer el total de las primitivas geométricas. Sin embargo, una de las principales desventajas de este método es que utiliza un ajuste por mínimos cuadrados, el cual es muy susceptible a errores inducidos por puntos atípicos.

En [41, 34] los autores presentan una metodología para detectar primitivas geométricas

buscando primero las primitivas de mayor tamaño y avanzando hasta detectar las más pequeñas. La metodología emplea información de los vértices de los puntos e información de los vectores normales asociados a cada uno de ellos. El enfoque requiere del cálculo de valores propios de los puntos y sus normales para determinar si un punto pertenece a una cierta primitiva. El método es inviable para conjuntos de datos muy grandes debido al cálculo de sus vectores y valores propios en cada iteración.

3.1.3 Trabajos que emplean MAC

En [56] se publicó un trabajo que permite segmentar varias primitivas geométricas como: planos, esferas, cilindros, conos y toroides. Los autores emplean MAC, un muestreo no uniforme, sólo manejan datos sin ordenamiento, requieren conocer los vértices de cada punto y los vectores normales de cada punto, pero el cálculo de los vectores normales de una nube de puntos es en si un problema difícil de resolver [66] además de producirse mucho errores en su estimación.

3.2 Algoritmo Muestra Aleatoria de Consenso (MAC)

El propósito de estimar los paramétros de un objeto matemático a un conjunto de datos observados, es encontrar la mejor aproximación posible de las ecuaciones que definen el modelo tomando como referencia los datos disponibles. Los métodos empleados para resolver este problema se pueden categorizar en dos: métodos directos y métodos que requieren de un proceso iterativo para encontrar un solución cercana a valores óptimos. En el área de la estadística se han desarrollado algunos métodos para resolver el problema de la estimación de parámetros mediante técnicas iterativas como por ejemplo: $estimador - M$, $estimador - L$, $estimador - R$ [46] y mínima media de cuadrados [51] entre otros. Estos enfoques permiten la regresión de parámetros como un problema de minimización del error, similar al método de mínimos cuadrados. La principal desventaja de estos métodos reside en el uso de complejas funciones no lineales que requieren de programas especiales

para obtener una solución. Como una alternativa en la estimación de parámetros se podría emplear un proceso iterativo combinado con un método directo.

MAC es un algoritmo iterativo, simple y fácil de implementar. No se requiere de métodos complejos de optimización y tampoco requiere gran capacidad de cómputo. La simpleza y eficiencia de trabajar de este algoritmo reside en la selección aleatoria de muestras para generar un modelo de parámetros. Posteriormente, el modelo es verificado con los datos observados hasta que se va mejorando la estimación de los parámetros en cada iteración. Diferentes variaciones de MAC han sido publicadas en la literatura [44, 60, 73] las cuales se catalogan de acuerdo a la precisión, robustez y eficiencia de este algoritmo.

3.2.1 Descripción del algoritmo MAC

Los principales pasos realizados por el algoritmo MAC se describen en los siguientes puntos:

1 Selección aleatoria de un subconjunto h del conjunto total de puntos dados. La cardinalidad de este conjunto de datos h depende de la mínima cantidad de datos que se requieren para generar un modelo lineal que es fácil y rápido de calcular.

2 A partir del subconjunto de datos en h del paso 1 se genera un modelo; es decir, se estiman los parámetros del modelo.

3 Por cada dato en el conjunto total se aplica la siguiente regla: si la distancia del punto observado al modelo es menor o igual que ϵ, entonces el punto es aceptado como un dato que pertenece al modelo. De lo contrario, se considera como un punto atípico y es rechazado. A ϵ se le conoce como el umbral de aceptación del modelo y está definido por la naturaleza de los datos observados.

4 Si el modelo de parámetros en la iteración actual es mejor que el anterior (esto es, el modelo actual tiene más puntos aceptados) entonces se actualiza el modelo de

parámetros por el mejor. De lo contrario se conserva el modelo anterior.

5 Se repiten los pasos 1 al 4 hasta que se cumple un cierto número de iteraciones.

El algoritmo retorna un subconjunto de datos que se ajustaron mejor al modelo final y un subconjunto de datos que están fuera del umbral del algoritmo. El algoritmo MAC ha mostrado tener buenos resultados con conjuntos de datos hasta con 50 % de puntos atípicos y ruido [18]. En la figura 3.1 se ilustra gráficamene la operación de este algoritmo. Aunque en los datos hay ruido y puntos atípicos, el algoritmo puede encontrar los parámetros de la recta que mejor se aproximan a los datos.

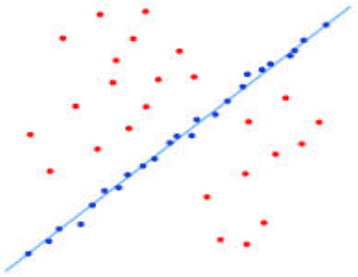


Figura 3.1: Ajuste de una recta a un conjunto de puntos observados mediante el algoritmo MAC

3.2.2 Complejidad del algoritmo MAC

La complejidad del algoritmo MAC está dominada por dos principales factores: el número mínimo de iteraciones k y el costo de construir el modelo candidato.

3.2.2.1. Máximo número de iteraciones para determinar un modelo

La decisión de detener la selección de datos individualmente, para generar nuevos modelos que mejor se ajusten al total de los puntos, se puede limitar a la cantidad de

iteraciones k, necesarias para seleccionar un conjunto de tamaño n que generen un modelo que mejor ajusten el total de datos.

La máxima cantidad de subconjuntos que se pueden generar de un total de N puntos, seleccionando n puntos necesarios para construir un modelo candidato, es $k = \binom{N}{n}$, lo cual puede ser un valor muy grande incluso para valores pequeños de N. El valor de k depende del número de puntos que pertenecen a un modelo geométrico del total de puntos N. Si Y representa los puntos que cumplen la tolerancia de ser aceptados por el modelo (puntos típicos), entonces la ecuación (3.1) define la probabilidad de tomar un punto del modelo correcto:

$$w = \frac{Y}{N} \tag{3.1}$$

entonces la probabilidad de seleccionar aleatoriamente en un mismo tiempo los n puntos pertenecientes al modelo es: w^n. Se define z como la probabilidad de que al menos un conjunto contiene únicamente puntos típicos, y se define a z en función de w, k y n como se muestra en la ecuación (3.2).

$$z = 1 - (1 - w^n)^k \tag{3.2}$$

La ecuación (3.3) nos da un valor estimado del valor de k en función de z, w y n:

$$k = \frac{\log(1 - z)}{\log(1 - w^n)} \tag{3.3}$$

Al parámetro z se le conoce también como nivel de confianza o nivel de éxito de que en k iteraciones se obtenga un subconjunto de puntos típicos. La tabla 3.1 muestra algunos resultados de evaluar la ecuación (3.3) para w=0.5 (50 % de puntos típicos) con z= 0.95 y z= 0.5.

Tabla 3.1: Valores de k para diferentes valores de n con z=0.95 y z=0.5, empleando la ecuación (3.3)

	k	
n	z=0.95	z=0.5
2	11	3
3	23	6
4	47	12
5	95	23
6	191	45
7	382	89
8	766	178
9	1,533	356

3.3 Descomposición de matrices en Valores Singulares

En álgebra lineal, la Descomposición en Valores Singulares (DVS) (en inglés *Singular Value Decomposition (SVD)*) de una matriz real o compleja es una factorización de la misma. Esta descomposición es considerada como una generalización del problema de vectores y valores propios.

3.3.1 Descripción de la Descomposición en Valores Singulares

El teorema de la DVS [24, 25] se define de la siguiente forma: Sea A una matriz de tamaño $m \times n$ con $m \geq n$, A puede ser factorizada como:

$$A = U\Sigma V^T \tag{3.4}$$

donde $U^TU = V^TV = VV^T = I_n$ y $\Sigma = \text{diag}(\lambda_1, ..., \lambda_n)$.

La matriz U consiste de n vectores propios[1] ortonormalizados asociados con los n valores propios de AA^T. La matriz V consiste de los vectores propios ortonormalizados de A^TA. Los elementos en la diagonal de Σ son las raíces cuadradas no negativas de los valores propios de A^TA y son conocidos como valores singulares [25].

Podemos resumir la factorización DVS de la forma siguiente:

- U, vectores singulares izquierdos de A son un conjunto de vectores propios de AA^T, matriz ortonormal de tamaño $m \times m$.
- V^T, vectores singulares derechos de A son un conjunto de vectores propios de A^TA, matriz ortonormal de tamaño $n \times n$.
- Σ, matriz diagonal con las raíces cuadradas de los valores singulares no negativos de A^TA, tamaño $m \times n$.

Para entender el funcionamiento del cálculo de DVS consideremos el siguiente ejemplo:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

A es una matriz rectangular de 3×2. Primero se calcularán los valores de la matriz V y de la matriz Σ a partir de la matriz A^TA.

$$A^TA = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = W$$

El vector propio $\mathbf{x}$ de la matriz W es una matriz de $n \times 1$ tal que $W\mathbf{x} = \lambda\mathbf{x}$, donde λ es un valor escalar real que recibe el nombre de valor propio.

[1] También conocidos como autovectores o eigenvectores

La alternativa obvia sería el vector propio cero. Este último se descarta ya que la condición $W\mathbf{x} = \lambda\mathbf{x}$ se cumpliría para cualquier valor de λ teniendo, entonces, un número infinito de soluciones para λ.

Con la finalidad de encontrar el valor de λ, se analiza la expresión $W\mathbf{x} = \lambda\mathbf{x}$ de la cual $\lambda\mathbf{x}$ se puede escribir como $(\lambda I)\mathbf{x}$ (I representa la matriz identidad) y por lo tanto se puede re-escribir como:

$$W\mathbf{x} = \lambda\mathbf{x}$$

$$W\mathbf{x} - \lambda I\mathbf{x} = 0$$

$$(W - \lambda I)\mathbf{x} = 0 \tag{3.5}$$

Esta última representa un sistema homogéneo con n ecuaciones que tiene al menos la solución trivial. La única manera de que tenga soluciones no triviales es que:

$$\det(W - \lambda I) = 0 \tag{3.6}$$

sustituyendo los valores en la ecuación (3.6):

$$\det\left(\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) = \det\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}$$

haciendo las operaciones:

$$(2-\lambda)(2-\lambda) - 1 = \lambda^2 - 4\lambda + 3 = 0$$

resolviendo la ecuación cuadrática para λ se obtiene λ_1=3 y λ_2=1. Sustituyendo los valores propios en la ecuación (3.5) se pueden definir los vectores propios asociados. Para λ_1=3:

$$\begin{bmatrix} 2-3 & 1 \\ 1 & 2-3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$-x_1 + x_2 = 0$$

$$x_1 - x_2 = 0$$

Si del conjunto de ecuaciones tomamos: $x_2 = x_1$, si x_1=1, entonces x_2=1

el primer vector es:

$$v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

pero se necesita que sean unitarios, por lo cual se divide por la norma del vector:

$$v_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}$$

Para λ_2=1 sustituyendo en la ecuación (3.5) se obtiene v_2:

$$\begin{bmatrix} 2-1 & 1 \\ 1 & 2-1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$x_1 + x_2 = 0$$

$$x_1 + x_2 = 0$$

Si del conjunto de ecuaciones tomamos: $x_2 = -x_1$, si x_1=1, entonces x_2=-1 y el segundo vector normalizado queda:

$$v_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 0.7071 \\ -0.7071 \end{bmatrix}$$

concatenando v_1 y v_2 obtenemos la matriz V:

$$V = \begin{bmatrix} v_1 | v_2 \end{bmatrix} = \begin{bmatrix} 0.7071 & 0.7071 \\ 0.7071 & -0.7071 \end{bmatrix}$$

Y la matriz Σ son las raíces cuadradas de los valores singulares de $A^T A$:

$$\Sigma = \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{1} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1.7371 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Para obtener los valores de la matriz U se desarrolla la misma técnica obteniendo:

$$U = \begin{bmatrix} 0.4082 & 0.7071 & 0.5773 \\ 0.4082 & 0.7071 & 0.5773 \\ 0.8165 & -0.111 & -0.5773 \end{bmatrix}$$

Para comprobar que el cálculo es correcto se puede hacer la multiplicación de la factorización:

$$A = U\Sigma V^T = \begin{bmatrix} 0.4082 & 0.7071 & 0.5773 \\ 0.4082 & 0.7071 & 0.5773 \\ 0.8165 & -0.111 & -0.5773 \end{bmatrix} \begin{bmatrix} 1.7371 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0.7071 & 0.7071 \\ 0.7071 & -0.7071 \end{bmatrix}$$

$$A = U\Sigma V^T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

3.3.2 Complejidad de la Descomposición en Valores Singulares

La complejidad temporal del cálculo de DVS está dominada por el valor de m y n. En la tabla 3.2 se muestran las complejidades estimadas para el paquete de software LAPACK [45] obtenidas de [24] y están dadas según las matrices que se desean obtener de la DVS. En algunas ocasiones sólo se requieren los valores singulas Σ, o Σ y V [12]. Este paquete de álgebra lineal es utilizado en la mayoría de las librerías de lenguajes de programación como Python, Matlab, Octave, entre otros.

Tabla 3.2: Complejidades reportadas por [24] para el cálculo de DVS

Requerido	**Complejidad**
Σ	$4mn^2 - 4n^3/3$
Σ, V	$4mn^2 + 8n^3$
Σ, U	$4m^2n - 8mn^3$
Σ, U, V	$4m^2n + 8mn^2 + 9n^3$

3.4 Estimación de primitivas geométricas

En esta sección se describe la generación de los modelos de primitivas geométricas a partir de n puntos. Estos modelos se emplearon para encontrar primitivas en una nube de puntos dada.

3.4.1 Plano

El plano se estimó de la misma forma que se describió en la sección 2.3.1, en la página 23. La estimación del plano requiere encontrar las matrices Σ y V de la descomposición en valores singulares de una matriz de tamaño 3×3.

3.4.2 Esfera

La esfera se estimó de la misma forma que se describió en la sección 2.3.2, en la página 28.

3.4.3 Cilindro

Para generar un modelo de consenso para el cilindro [30] se requieren al menos 9 puntos sobre la superficie del cilindro. El cilindro cumple la restricción de que existe un plano de tal forma que todos los puntos del cilindro se pueden proyectar en tal plano y son co-circulares. Si ese plano, sin pérdida de generalidad, es el plano xy, esta restricción puede expresarse como:

$$x^2 + y^2 - r^2 = 0, \tag{3.7}$$

donde r es el radio del cilindro. Esta es justamente la ecuación del círculo que en notación matricial se puede re-escribir como:

$$\mathbf{x}^\mathsf{T} C' \mathbf{x} = 0 \tag{3.8}$$

donde $\mathbf{x} = [x, y, z, 1]^\mathsf{T}$ y la matriz C' es igual a:

$$C' = \begin{bmatrix} D & \mathbf{0} \\ \mathbf{0}^\mathsf{T} & -r^2 \end{bmatrix},$$

con la matriz D igual a:

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.9}$$

Debido a que la orientación del plano de proyección es en general desconocida, es necesario aplicar una rotación y una translación al cilindro de forma:

$$M = \begin{bmatrix} I & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

El cilindro general será entonces igual a $C = M^{-\mathsf{T}} C' M^{-1}$. Desarrollando esta expresión tenemos:

$$C = \lambda \begin{bmatrix} RDR^{\mathsf{T}} & -RDR^{\mathsf{T}}\mathbf{t} \\ -\mathbf{t}^{\mathsf{T}}RDR^{\mathsf{T}} & \mathbf{t}^{\mathsf{T}}RDR^{\mathsf{T}}\mathbf{t} - r^2 \end{bmatrix} = \lambda \begin{bmatrix} E & \mathbf{q} \\ \mathbf{q}^T & q \end{bmatrix} \tag{3.10}$$

La matriz E es simétrica por lo que tiene seis elementos distintos, el vector $\mathbf{q}$ tiene tres variables y q es un escalar. En total, la matriz C, por ser simétrica, tiene 10 grados de libertad. Pero la matriz C es similar hasta un factor de escala λ, por lo que sólo tiene nueve grados de libertad.

Con 9 puntos $\mathbf{x}_i$, para $i = 1, \ldots, 9$, sobre la superficie del cilindro, es necesario encontrar los 10 valores distintos para la matriz C. Si estos elementos de la matriz C se colocan en forma de un un vector $\mathbf{c}$ como:

$$\mathbf{c} = [c_{11}, c_{12}, c_{13}, c_{14}, c_{22}, c_{23}, c_{24}, c_{33}, c_{34}, c_{44}]^{\mathsf{T}}.$$

Entonces, a partir de la ecuación (3.8) se encuentra una matriz A, para formar un sistema de ecuaciones $A\mathbf{c} = 0$, y cada renglón de esta matriz será igual a:

$$[x_i^2, 2x_iy_i, 2x_iz_i, 2x_i, y_i^2, 2y_iz_i, 2y_i, z_i^2, 2z_i, 1]$$

La solución para $A\mathbf{c} = 0$ es el vector singular asociado su menor valor singular. Esta solución estima $\|A\mathbf{c}\|$ sujeto a la restricción $\|\mathbf{c}\| = 1$.

Una vez que se tiene la matriz C, los valores necesarios se calculan primero a partir

de $\lambda E = RDR^{\mathsf{T}}$. Note que esto es la eigendescomposición de la matriz E. Entonces se realizan los siguientes pasos para obtener los valores para R, $\mathbf{t}$ y r:

1. Se realiza entonces primero la eigendescomposición de la matriz E'. Esta matriz se forma con los primeros tres renglones y tres columnas de la matriz C. Entonces se tiene $E' = RFR^{\mathsf{T}}$. Aquí ya se obtiene R.

2. La matriz F es singular y tiene dos eigenvalores idénticos, entonces el factor de escala se calcula como λ=$(f'_{11} + f'_{22})/2$.

3. Se divide C entre el valor de este factor de escala, y se obtiene $\mathbf{q}$.

4. $\mathbf{t}$ se calcula como $\mathbf{t} = -E^{-1}\mathbf{q}$.

5. Finalmente el radio se obtiene de

$$q = \mathbf{t}^{\mathsf{T}} RDR^{\mathsf{T}}\mathbf{t} - r^2, \text{ y}$$
$$r^2 = \mathbf{t}^{\mathsf{T}} RDR^{\mathsf{T}}\mathbf{t} - q$$

3.4.4 Estimación lineal del cono

La ecuación del cono de base circular sobre el plano xy se puede definir por la expresión $c^2(x^2+y^2) = z^2$, o reescrita como $z^2 - c^2(x^2+y^2) = 0$. Aquí c^2 es simplemente un factor de escala para controlar la forma del cilindro. Esta última expresión en forma matricial es:

$$\mathbf{x}^{\mathsf{T}} D'\mathbf{x} = 0 \tag{3.11}$$

con

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \text{y} \quad D' = \begin{bmatrix} -c^2 & 0 & 0 \\ 0 & -c^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Si se aplica una rotación y traslación al cono, como se hizo en el caso del cilindro, se obtiene:

$$D = \lambda \begin{bmatrix} R & 0 \\ -\mathbf{t}^\mathsf{T} R & 1 \end{bmatrix} \begin{bmatrix} D' & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} R^\mathsf{T} & -R^\mathsf{T}\mathbf{t} \\ 0 & 1 \end{bmatrix}$$

$$D = \lambda \begin{bmatrix} RD'R^\mathsf{T} & -RD'R^\mathsf{T}\mathbf{t} \\ -\mathbf{t}^\mathsf{T} RD'R^\mathsf{T} & \mathbf{t}^\mathsf{T} RD'R^\mathsf{T}\mathbf{t} \end{bmatrix}$$

Para estimar esta matriz D se necesitan al menos 9 puntos sobre la superficie del cono. D es simétrica y si ordenamos sus elementos desconocidos en el vector:

$$\mathbf{d} = [d_{11}, d_{12}, d_{13}, d_{14}, d_{22}, d_{23}, d_{24}, d_{33}, d_{34}, d_{44}]^\mathsf{T}.$$

Se necesita construir el sistema de ecuaciones homogéneo $A\mathbf{d} = 0$. La matriz A se calcular a partir de $\mathbf{x}^\mathsf{T} D\mathbf{x} = 0$ y cada renglón es igual a:

$$[x_i^2, 2x_i y_i, 2x_i z_i, 2x_i, y_i^2, 2y_i z_i, 2y_i, z_i^2, 2z_i, 1]$$

para un punto $[x_i, y_i, z_i]^\mathsf{T}$ dado.

Se encuentra la estimación para $\mathbf{d}$ usando la DVS ($\mathbf{d}$ es el vector singular asociado al menor valor singular de A).

Una vez que se tiene la matriz D, realizando los siguientes pasos se obtienen los parámetros del cono:

1. Si F es la matriz de los primeros tres renglones y tres columnas de la matriz D, se realiza su eigendescomposión como $F = RGR^\mathsf{T}$. Aquí se obtiene R.

2. El factor de escala es igual al valor del último eigenvalor en la matriz diagonal G.

3. $D' = G/\lambda$ y aquí se obtiene el valor para c: $c = \sqrt{-d_{11}}$

4. Tomando $\mathbf{q} = -RD'R^T\mathbf{t}$, entonces $\mathbf{t} = -RD'^{-1}R^T\mathbf{q}$ con

$$D'^{-1} = \begin{bmatrix} -1/c^2 & 0 & 0 \\ 0 & -1/c^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3.5 Modelos geométricos propuestos para guardar y representar primitivas 3D

En la sección previa se describió la forma de generar los modelos candidatos para la extracción de primitivas geoméricas en nubes de puntos. Posterior al trabajo de segmentación de primitivas, al aplicar la metodología propuesta con el algoritmo MAC se tienen un conjunto de primitivas y los parámetros encontrados para cada primitiva. Con esta información se propone un modelo final que pueda ser utilizado en diferentes herramientas de software de graficación como: CAD, OpenGL, GNU-Plot, Matlab, Povray, etc. Los modelos presentados en esta sección son ligeramente diferentes a los parámetros obtenidos en la estimación de primitivas, dado que algunas herramientas requieren información adicional no conocida en la extracción de datos pero que se pueden obtener con la información procesada.

3.5.1 Plano

El borde de un plano puede ser triangular definido por tres vértices, o rectangular definido por cuatro vértices y polígonal con más de cuatro vértices. En este trabajo nos interesamos en encontrar y definir los planos que se pueden definir por cuatro puntos, por lo cual se propone el modelo para plano de la forma $plano = \{p_1, p_2, p_3, p_4\}$. Estos cuatro vértices son seleccionados del conjunto de puntos de la primitiva tomando p_1 y p_2 que son los puntos con la mayor distancia de separación. Para seleccionar los puntos p_3 y p_4

se calcula la máxima distancia de los puntos restantes a la recta formada por p_1 y p_2.

3.5.2 Esfera

Para la esfera es más sencillo definir su modelo, dado que una esfera es invariante a traslación y rotación. Su modelo lo definimos como $esfera = \{\mathbf{r}_c, r\}$, donde $\mathbf{r}_c = (x_c, y_c, z_c)$ es el centro y r es el radio.

3.5.3 Cilindro

Para el modelo del cilindro se seleccionó la notación: $cilindro = \{p_i, p_f, r\}$, donde p_i y p_f son los puntos inicial y final respectivamente del eje director del cilindro y r es el radio del cilindro. Los puntos p_i y p_f son seleccionados del conjunto de datos del cilindro como aquellos puntos más distantes entre sí, y posteriormente estos puntos son proyectados sobre el eje del cilindro para obtener las coordenadas inicial y final del largo del cilindro.

3.5.4 Cono

El modelo para el cono se definió de la forma: $cono = \{p_i, r_i, p_f, r_f\}$ donde p_i y p_f son los puntos inicial y final del eje del cono calculados de forma idéntica al modelo del cilindro. r_i y r_f son el radio inicial y el radio final del cono, respectivamente. Estos valores se obtienen tomando la distancia ortogonal de los dos puntos más distante entre sí hacia el eje del cono.

3.6 Extracción de primitivas geométricas 3D

En esta sección se presentan dos metodologías para la extracción de primitivas geométricas empleando el paradigma MAC y la generación de modelos descritos en la sección 3.4. En la sección 3.6.2 se describe un algoritmo para detectar múltiples primitivas y su reconstrucción. En la sección 3.7 se describe una metodología para mejorar la convergencia de

MAC considerando que los datos están ordenados, es decir, están estructurados de forma contigua a sus vecinos más cercanos.

3.6.1 Estimación de iteraciones k para estimar los parámetros de cuatro primitivas en una nube de puntos

En este trabajo se propone emplear el algoritmo MAC para estimar los parámetros de las ecuaciones que describen las primitivas geométricas como: planos, esferas, cilindros y conos. En la tabla 3.3 se muestra la cantidad de puntos mínimos necesarios para estimar los parámetros de cada modelo de primitiva geométrica empleados en este trabajo.

Tabla 3.3: Puntos mínimos necesarios para construir un modelo de parámetros de primitivas geométricas

Primitiva geométrica	**Cantidad mínima de puntos**
Plano	3
Esfera	4
Cilindro	9
Cono	9
Promedio con redondeo al próximo entero superior	7

Una aproximación que se hizo en este trabajo para determinar la cantidad de iteraciones k necesarias para la convergencia de MAC, fue tomar un promedio de los puntos mínimos de la tabla 3.3 para poder emplear la ecuación (3.3).

Vamos a analizar el número de iteraciones necesarias para extraer cuatro primitivas en m puntos. Para extraer la primera primitiva tenemos $m/4$ puntos típicos y $3m/4$ puntos atípicos. Estamos considerando entonces que una cuarta parte de los puntos forman la primera primitiva y los restantes pertenecen a las otras primitivas. La relación de puntos típicos en puntos totales es:

$$w_1 = \frac{\frac{m}{4}}{\frac{m}{1}} = \frac{m}{4m} = \frac{1}{4}$$

y el número de iteraciones será igual a:

$$k_1 = \frac{\log(1-0.95)}{\log(1-w_1^7)} = \frac{\log(0.05)}{\log(1-0.25^7)} = 49,080.58 \rightarrow 49,081$$

Para extraer la segunda primitiva, ahora quitamos los puntos ya detectados de la primera primitiva como $m_2 = m - m/4 = 3m/4$. De éstos, un tercio son puntos típicos y 2/3 son puntos atípicos. La relación entre puntos típicos y puntos totales es:

$$w_2 = \frac{\frac{m_2}{3}}{\frac{m_2}{1}} = \frac{m_2}{3m_2} = \frac{1}{3}$$

y el número de iteraciones para encontrar la segunda primitiva es:

$$k_2 = \frac{\log(1-0.95)}{\log(1-w_2^7)} = \frac{\log(0.05)}{\log(1-0.33333^7)} = 6,550.17 \rightarrow 6,551$$

Para extraer la tercera primitiva, ahora $m_3 = m_2 - m_2/3 = 2m_2/3$, m_3 es el número de puntos que quedan para extraer la tercera primitiva, en esta situación la relación entre puntos típicos y puntos totales es:

$$w_3 = \frac{\frac{m_3}{2}}{\frac{m_3}{1}} = \frac{m_3}{2m_3} = \frac{1}{2}$$

y el número de iteraciones para extraer la tercera primitiva es:

$$k_3 = \frac{\log(1-0.95)}{\log(1-w_3^7)} = \frac{\log(0.05)}{\log(1-0.5^7)} = 381.95 \rightarrow 382$$

Para extraer la cuarta primitiva consideramos que un 10 % de los puntos que quedan son atípicos, por lo que la relación de puntos típicos a puntos totales es:

$$w_4 = \frac{m_3 - 0.1m_3}{m_3} = \frac{0.9m_3}{m_3} = 0.9$$

y el número de iteraciones para extraer la cuarta primitiva es:

$$k_4 = \frac{\log(1-0.95)}{\log(1-w_4^7)} = \frac{\log(0.05)}{\log(1-0.9^7)} = 4.6 \rightarrow 5$$

En la tabla 3.4 se muestra el número de iteraciones del algoritmo MAC para extraer cuatro primitivas geométricas propuestas en este trabajo a un valor de confianza $z = 0.95$. En la figura 3.2 se muestra el comportamiento del algoritmo MAC para detectar cuatro figuras geométricas a distintos niveles de confianza z, las coordenadas en el eje y están en unidades logarítmicas.

Tabla 3.4: Iteraciones k necesarias para extraer cuatro primitivas de una nube de puntos empleando el algoritmo MAC

Primitiva	k	Σk
Primera	49,081	49,081
Segunda	6,551	55,632
Tercera	382	56,014
Cuarta	5	56,019

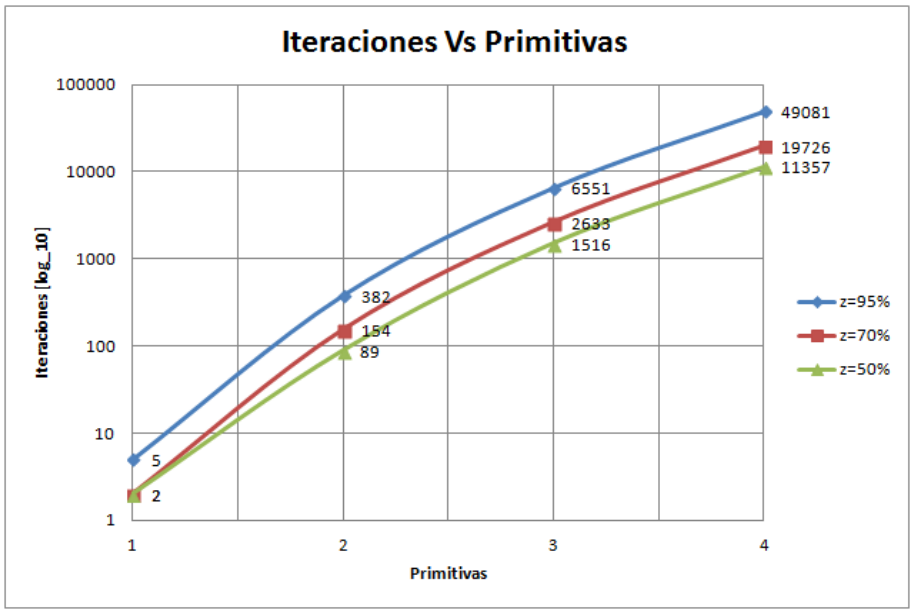


Figura 3.2: Gráfica de complejidad de MAC para extraer cuatro primitivas de una nube de puntos con diferentes niveles de confianza z

3.6.2 Primera metodología propuesta para la segmentación de primitivas empleando MAC

En esta sección se presenta un primer enfoque desarrollado para extraer primitivas geométricas de archivos de nubes de puntos. La metodología propuesta para la extracción de primitivas 3D se resume en los siguientes pasos:

1 Inicializar un conjunto vacío de primitivas segmentadas.

2 Generar un conjunto de modelos candidatos.

3 Evaluar candidatos con los puntos.

4 Buscar el candidato con el conteo de puntos más alto.

5 Si el modelo actual es mejor que el modelo obtenido previamente actualizar modelo con el nuevo modelo.

6 Repetir los pasos 2 a 5 en una área de la nube de puntos definida por una estimación mínima de tamaño de primitiva

7 Remover datos de la nube de puntos que ajustaron mejor a algún modelo generado.

8 Repetir pasos 2 a 7 hasta que toda la nube de puntos sea analizada.

9 Regresar un conjunto de primitivas segmentadas con sus parámetros y un subconjunto de puntos sobrantes que no pertenecen a ninguna primitiva.

Para mejorar la operación de la metodología propuesta se empleó una búsqueda localizada de primitivas, Myatt $et\ al.$ [11] proponen el uso de un muestreo no uniforme pero localizado para mejorar el proceso de extracción de modelos, y los autores muestran que el uso de estas técnicas ayudan a mejorar la probabilidad de encontrar un buen conjunto de datos que generen un modelo dentro de los datos. La estrategia de muestreo funciona

tomando inicialmente de forma uniforme un punto del total del área explorada y posteriormente se construye una hiperesfera alrededor de este punto con radio r. Los puntos faltantes para generar el modelo son tomados de manera uniforme dentro de la hiperesfera construida. El valor de r es un múltiplo de la media de las distancias entre puntos. El valor de r podría ser tomado de un diseño de experimentos, aunque los autores en [11] sugieren un valor de $r = 50$ para conjuntos de datos muy grandes. Para nuestra experimentación, este valor fue ajustado para el muestreo no uniforme.

3.6.3 Resultados, reconstrucción y visualización

Para evaluar la metodología propuesta en esta sección se analizaron tres nubes de puntos generados por el escáner, las nubes de puntos fueron tomadas de escenas donde había objetos compuestos principalmente de primitivas geométricas. La metodología fue implementada en el lenguaje de progamación Python (versión 2.7.6 64-bits) y la experimentación se realizó empleando un computadora con procesador Intel i5 a una velocidad de 2.5GHz, 4 GB de memoria RAM y sistema operativo Windows 7 de 64 bits. Para medir estadísticamente el comportamiento de nuestra metodología se ejecutó 11 veces el programa para cada nube de puntos. La información del tamaño de la nube de puntos, el valor de ϵ, el tiempo de ejecución y la cantidad de primitivas segmentadas se resumen en la tabla 3.5.

Tabla 3.5: Información de los conjuntos de datos empleados para la experimentación de la metodología propuesta.

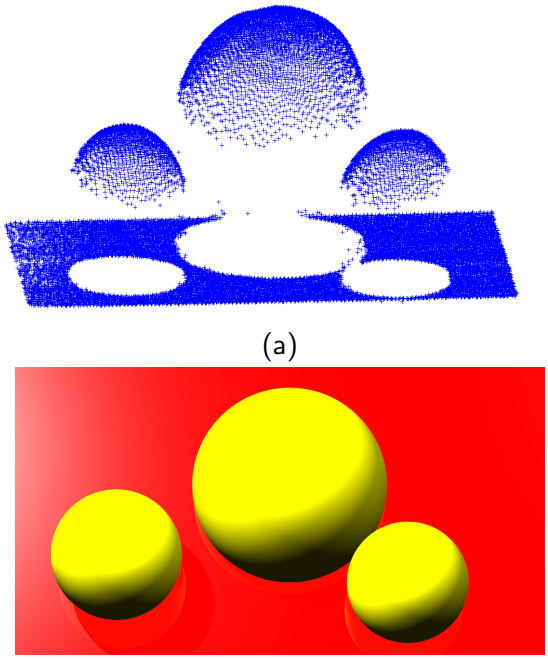
Conjunto de datos	**Puntos**	ϵ	**tiempo[seg.]**	**Primitivas**
esferas.dat	22,145	0.03	7.66	4
cilindro.dat	17,332	0.03	20.89	5
adaptadorUSB.dat	7,332	0.03	10.24	5

Para reconstruir la escena original se realizó un paso adicional empleando el software GSC de Povray empleando solamente operadores booleanos de unión. Para reconstruir la escena es posible tomar como referencia el objeto con mayor tamaño y la transformación

descrita por $S{=}R^{T}_{objetoPrincipal}{\cdot}R_i$, y una traslación $\mathbf{v}{=}R^{T}_{objetoPrincipal}{\cdot}(\mathbf{t}_i{-}\mathbf{t}_{objetoPrincipal})$, donde i son cada una de las primitivas restantes.

Las figuras 3.3 a la 3.5 muestran la graficación de los puntos 3D y su recreación mediante GSC de Povray.

La implementación es un método iterativo y el tiempo de ejecución depende del número de iteraciones. En estos experimentos se tuvo un buen desempeño con 1,000 iteraciones.



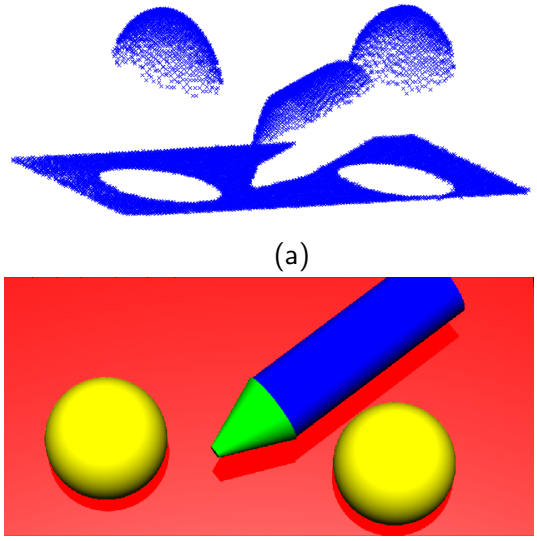
(a)

(b)

Figura 3.3: (a). Gráfica de la nube de puntos "esferas.dat". (b). Reconstrucción con GSC del conjunto de datos "esferas.dat" después de la segmentación de sus primitivas

3.7 Aceleración de la convergencia del algoritmo MAC considerando ordenamiento de datos

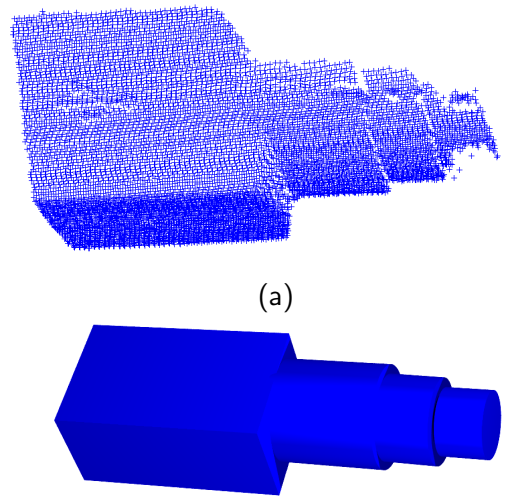
En general, el algoritmo MAC opera sin información sobre los datos. Durante una cantidad de iteraciones k el algoritmo hace un muestreo uniforme sobre el total de datos y ajusta los datos al mejor modelo estimado. En esta sección se presenta una comparación



(a)

(b)

Figura 3.4: (a). Gráfica de la nube de puntos "cilindro.dat". (b). Reconstrucción con GSC del conjunto de datos "cilindro.dat" después de la segmentación de sus primitivas



(a)

(b)

Figura 3.5: (a). Gráfica de la nube de puntos "adaptadorUSB.dat". (b). Reconstrucción con GSC del conjunto de datos "adaptadorUSB.dat" después de la segmentación de sus primitivas

del algoritmo MAC con muestreo uniforme y una mejora que se propuso emplear en este trabajo para acelerar la convergencia del algoritmo, tomando en consideración que los datos de entrada están ordenados. Es decir, cada punto en el arreglo de datos tiene en

sus posiciones contiguas a sus vecinos más cercanos.

En el propuesta para acelerar MAC, además de que los puntos están ordenados, se utilizan pocos puntos, sólo dos líneas de puntos escaneadas a la vez y no todos los puntos para detectar primitivas. Esto reduce dramáticamente el tiempo de ejecución de MAC.

3.7.1 Metodología propuesta para acelerar la convergencia del algoritmo MAC

Algoritmo 1 Algoritmo para extracción de primitivas geométricas empleando MAC y muestreo uniforme

Entrada: Nube de puntos P
Entrada: MAXITER (máxima cantidad de iteraciones)
$S \leftarrow \emptyset$
$k \leftarrow 0$
repetir
Seleccionar con muestreo uniforme h_m puntos de P para generar m modelos candidatos
Con h_m generar modelos H_m (plano, esfera, cilindro y cono)
para i $= 1$ **to** $|P|$ **hacer**
si $distancia(P_i, H_m) \leq \epsilon$ **entonces**
aceptar en h_m
sino
Descartar
fin si
fin para
Seleccionar mejor primitiva con $|h_m| \geq 0.15 \cdot |P|$
Si modelo actual mejor que anterior actualizar
$S \leftarrow H_m$
$n = contarPuntosSobrantes()$
si $n > 0.15 \cdot |P|$ **entonces**
Buscar nuevos modelos (ajustar plano, esfera, cilindro y cono)
fin si
hasta $n < 0.15 \cdot |P|$ o $k =$ MAXITER
regresa S

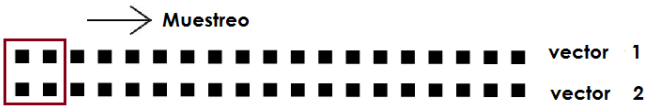


Figura 3.6: Estrategia de muestreo empleando nubes de puntos con datos ordenados

Algoritmo 2 Algoritmo empleando MAC para extraer primitivas geométricas considerando el ordenamiento de puntos

Entrada: Arreglos de datos ordenados
$P \leftarrow \emptyset$
$S \leftarrow \emptyset$
$p \leftarrow$ leer primeros 2 arreglos de datos
Ajustar datos empleando la estrategia de muestreo no uniforme para generar modelos de plano, esfera, cilindro y cono
$S \leftarrow H_m$
repetir
  $P \leftarrow p$
  $p \leftarrow$ leer siguientes 2 arreglos de datos
  Ajustar puntos a modelos previamente detectados
  Actualizar modelos
  $n = contarPuntosSobrantes()$
  si $n > 0.15 \cdot |P|$ **entonces**
    Buscar nuevos modelos empleando la estrategia de muestreo no uniforme para crear H_m modelos planos, esferas, cilindros y conos
    $S \leftarrow H_m$
  fin si
hasta No más arreglos de datos para leer o $n < 0.15 \cdot |P|$
regresa S

3.7.2 Experimentación y resultados

Los algoritmos 1 y 2 fueron implementados en el lenguaje de progamación Python (versión 2.7.6 64-bits), y se tomaron 20 diferentes archivos de nubes de puntos de prueba generados por el escáner láser desarrollado en este trabajo. Cada algoritmo fue ejecutado 21 veces con cada instancia de archivos de nubes de puntos; y fueron contabilizadas la cantidad de iteraciones requeridas por cada algoritmo para realizar la segmentación de primitivas. La experimentación se realizó empleando un computadora con procesador Intel i5 a una velocidad de 2.5GHz, 4 GB de memoria RAM y sistema operativo Windows 7 de

64 bits.

En las tablas 3.7 y 3.6 se muestran los resultados de la ejecución de los algoritmos 1 y 2 respectivamente. En estas tablas se reportan la cantidad de iteraciones que necesitó cada algoritmo para segmentar las primitivas geométricas contenidas en los archivos de nubes de puntos. En estas tablas la columna "Primitivas" indica el tipo de primitiva geométrica detectada: (p)lano, (e)sfera, (c)ilindro y (v) cono.

Tabla 3.6: Resultados estadísticos de las iteraciones necesarias para segmentar diferentes primitivas en diferentes nubes de puntos empleando MAC y considerando el ordenamiento de datos con la estrategia de muestreo no uniforme propuesto. Resultados de 21 ejecuciones. Las letras en paréntesis indican el tipo de primitiva: (p)lano, (e)sfera, (c)ilindro y (v) cono

$\|P\|$	**Promedio**	**Max**	**Min**	**Mediana**	**Desv. Estándar**	**Primitivas**
6310	196.57	197	196	197	0.51	p, e
7415	255.43	256	255	255	0.51	p, v
6626	155.38	156	155	155	0.50	p, c
9327	334.48	335	334	334	0.51	p, e, c, v
3155	2.43	3	2	2	0.51	p
3155	2.52	3	2	3	0.51	c
6585	160.48	161	160	160	0.51	$p, 2e$
7422	260.43	261	260	260	0.51	p, e, c
4686	75.71	76	75	76	0.46	p, c
5938	208.48	209	208	208	0.51	$p, 3e$
4669	88.57	89	88	89	0.51	p, e, v
6310	196.57	197	196	197	0.51	$2p$
6310	130.52	131	130	131	0.51	p, c
4750	79.52	80	79	80	0.51	$2p, c$
3188	16.52	17	16	17	0.51	e
5666	198.57	199	198	199	0.51	$2p, 2c$
5995	250.38	251	250	250	0.50	$4p$
4733	34.57	35	34	35	0.51	$2p, 2c$
6811	16.57	17	16	17	0.51	c
5366	113.62	114	113	114	0.50	p, c
Promedio:	**138.866**	**139.35**	**138.35**	**138.95**	**0.506**	

Considerando el promedio total de las iteraciones para las 20 instancias de datos de las tablas 3.7 y 3.6, se puede observar una mejora de 737 %. Esto nos permite hacer una conclusión: el ordenamiento de datos permite hacer una mejora en el muestreo de puntos

Tabla 3.7: Resultados estadísticos de las iteraciones necesarias para segmentar diferentes primitivas en diferentes nubes de puntos empleando MAC y sin considerar el ordenamiento de datos con la estrategia de muestreo no uniforme propuesto. Resultados de 21 ejecuciones. Las letras en paréntesis indican el tipo de primitiva: (p)lano, (e)sfera, (c)ilindro y (v) cono

$\|P\|$	**Promedio**	**Max**	**Min**	**Mediana**	**Desv. Estándar**	**Primitivas**
6310	1090.48	1400	67	1400	567.64	p, e
7415	1400.00	1400	1400	1400	0.00	p, v
6626	1400.00	1400	1400	1400	0.00	p, c
9327	1400.00	1400	1400	1400	0.00	p, e, c, v
3155	34.52	85	7	34	22.38	p
3155	32.71	77	9	29	19.53	c
6585	1400.00	1400	1400	1400	0.00	$p, 2e$
7422	1400.00	1400	1400	1400	0.00	p, e, c
4686	1338.29	1400	104	1400	282.81	p, c
5938	1400.00	1400	1400	1400	0.00	$p, 3e$
4669	1400.00	1400	1400	1400	0.00	p, e, v
6310	1076.10	1400	107	1400	529.65	$2p$
6310	1400.00	1400	1400	1400	0.00	p, c
4750	987.14	1400	80	1400	599.54	$2p, c$
3188	11.05	45	2.	7	12.01	e
5666	1400.00	1400	1400	1400	0.00	$2p, 2c$
5995	1400.00	1400	1400	1400	0.00	$4p$
4733	1400.00	1400	1400	1400	0.00	$2p, 2c$
6811	262.90	1400	5	54	482.78	c
5366	223.48	1400	4	11	449.59	p, c
Promedio:	**1023.9215**	**1202.05**	**789.8**	**1057.8**	**148.7425**	

para estimar primitivas geométricas.

4

Registro de nubes de puntos empleando descriptores de superficie

En la literatura, al problema de estimar una transformación T que minimice el error cuadrático entre un conjunto de datos en 3D en un sistema de coordenadas de referencia y un segundo conjunto de datos en 3D, en un sistema de coordenadas de referencia diferente, es conocido como: estimación de pose [68], registro [15], alineamiento [17] o emparejamiento de superficies [28], entre otros. En muchos de los problemas de visión por computadora se involucra el problema de registro de conjuntos de datos en 2D y 3D [19]. El registro entre conjuntos de datos es relevante en la integración de múltiples nubes de puntos obtenidas con escáneres láser, ingeniería inversa, reconocimiento de patrones y CAD, entre otros.

El algoritmo más popular para resolver este problema se denomina Puntos más Cercanos Iterativo (PMCI), (en inglés *Iterative Closest Point* (ICP)), debido a su fácil implementación y rápida convergencia [43, 72]. Sin embargo, la parte de mayor interés radica en cómo hacer más eficiente la representación de la información y la estimación de los

parámetros involucrados en la matriz de transformación.

En este capítulo se presenta una metodología que emplea algunas características del algoritmo PMCI y un optimizador de uso general como el propuesto por Levenberg-Marquardt [39]. El enfoque propuesto utiliza vectores que pueden ser empleados como descriptores de superficie; tales vectores se proponen para describir: planos, cilindros y conos. Adicionalmente, se emplean vértices de puntos 3D o puntos que describen el centro de una esfera como puntos de control para lograr el registro final de los datos. Utilizando la información de los puntos 3D y los vectores, se propone la optimización de una función objetivo mediante el algoritmo de Levenberg-Marquardt, dando mayor peso a la parte de ajuste de superficies. También se introduce una descripción del algoritmo PMCI, se proporciona una descripción del algoritmo Levenberg-Marquardt, así como la descripción de una técnica de agrupamiento y búsqueda de datos mediante una estructura tipo árbol. Adicionalmente, para verificar la metodología propuesta se hicieron diferentes pruebas con datos generados sintéticamente y con datos generados con el escáner desarrollado en este trabajo.

4.1 Algoritmo *Puntos más Cercanos Iterativo*

Desde la publicación del algoritmo PMCI, propuesto por Besl y McKay [43] e independientemente por Zhang [72], se han hecho muchas variantes de este algoritmo en función de cada uno los pasos que ejecuta [15, 55]. El algoritmo PMCI se compone básicamente de los siguientes cuatro pasos:

1 Buscar los puntos más cercanos entre las nubes de puntos de referencia y de prueba para establecer un conjunto de correspondencias.

2 Estimar un matriz de transformación T que disminuya el error cuadrático entre correspondencias.

3 Transformar los datos de la nube de puntos de prueba empleando la matriz de transformación T estimada en el paso 3.

4 Iterar (repetir pasos 1 a 3).

El algoritmo PMCI, primero trata de definir puntos de referencia que correlacionen datos en cada uno de los conjuntos de datos a registrar. Posteriormente, basado en las correspondencias encontradas, se trata de estimar una transformación geométrica que minimice el error cuadrático entre las correlaciones encontradas, es decir, que minimice la distancia entre estas correspondencias. Pero el buscar las correspondencias entre dos conjuntos de datos es un problema en sí dificil [27]. La técnica más utilizada para este paso es utilizar la mínima distancia Euclidiana. El segundo paso en el algoritmo es hacer la estimación de la matriz de transformación; dos de los métodos más populares y utilizados son el uso de cuaterniones [43, 72] y la descomposición en valores singulares [27]. Más recientemente, se ha usado también la minimización de una función de energía por mínimos cuadrados [19].

De los principales trabajos hechos, la mayoría trabaja directamente con puntos. Sin embargo, algunos autores han trabajado en agregar información al problema, para tratar de hacer converger más rápido el registro. Algunos han agregado información sobre vectores tangentes a una superficie [16], información sobre color, ajuste a planos, ajuste a algunas cuádricas y teselaciones[1] [15, 55, 53].

4.2 Árboles k-dimensionales

Se usaron los árboles k-dimensionales para ordenar las nubes de puntos y encontrar más eficientemente los puntos más cercanos en otra nube de puntos.

[1]Una teselación (también conocido como teselado) consiste en el recubrimiento de una superficie por medio de un patrón de figuras de tal forma que no exista ningún hueco entre una figura y otra, y que las figuras estén dispuestas sin superponerse unas sobre otras

Un árbol k-dimensional [49, 65, 4], abreviado como árbol-kd, es una estructura de datos que agrupa puntos de un espacio de k-dimensiones en secciones divididas por hiperplanos perpendiculares a uno de los eje de coordenadas. Cada nodo en el árbol representa un punto del conjunto de datos, por lo cual cada hiperplano pasa por alguno de los puntos. Cada nodo puede ser considerado como un hiperplano que divide el espacio en dos partes conocidas como subespacios. Los puntos a la izquierda y derecha de tal punto son después también divididos en nuevos subespacios hasta terminar de construir el árbol y agrupar los puntos.

La manera de seleccionar el nodo raíz para la construcción del árbol es mediante la selección del valor de la mediana de los datos en algunos de los ejes de coordenadas. Por ejemplo, si se selecciona primero el eje x, todos sus descendientes estarán alineados con tal eje. El siguiente eje podría ser y y todos sus descendientes estarán alineados a tal eje y así sucesivamente hasta completar las k diferentes dimensiones. En la figura 4.1 se ilustra la partición de un conjunto de datos en 2 dimensiones (k=2). En el ejemplo, primero se particiona por el eje x y posteriormente por el eje y, siguiendo la partición por la mediana de los datos contenidos en cada subespacio.

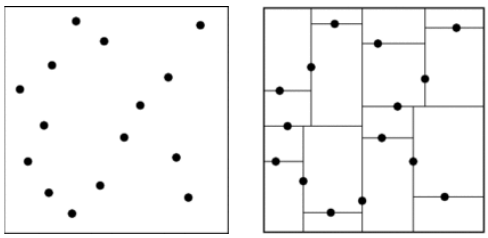


Figura 4.1: Se ilustra la partición de un conjunto de datos en 2 dimensiones (k=2). En el ejemplo, primero se particiona por el eje x y posteriormente por el eje y, siguiendo la partición por la mediana de los datos contenidos en cada subespacio

Las operaciones que se pueden hacer con este tipo de estructuras son: agregar un nodo, eliminar un nodo y buscar un nodo. Los árboles-kd son eficientes cuando la cantidad de dimensiones no es alta [49, 65, 4]. En la tabla 4.1 se muestran las diferentes complejidades computaciones encontradas en la literatura para esta estructura de datos.

Tabla 4.1: Complejidad temporal para las operaciones con árboles k-dimensionales. n es el número de puntos en el árbol.

Construcción	Insertar nodo	Buscar nodo
$\mathcal{O}(n \log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$

4.3 Algoritmo Levenberg-Marquardt

El algoritmo de Levenberg-Marquardt (LM) [39] es un método bien conocido para resolver iterativamente problemas no lineales de mínimos cuadrados incluyendo ajuste no lineal de datos [36, 32, 37]. LM trabaja como una combinación del método del descenso del gradiente y el método de Gauss-Newton. El algoritmo de LM puede encontrar una solución, en la mayoría de los casos, incluso si la solución inicial está lejos del mínimo global. La principal aplicación de LM es para resolver problemas en ajuste de curvas: dado un conjunto de datos de prueba d y un conjunto de datos independientes m, el objetivo es optimizar los parámetros en un vector $\mathbf{a}_i$ ($i = 1, 2, 3, ..., n$, n=card($\mathbf{a}$)), de la curva $f(d, \mathbf{a})$ tal que la suma de los errores al cuadrado sea mínima:

$$E(\mathbf{a}) = \sum_{j=1}^{n} [m_j - f(d_j, \mathbf{a})]^2 \tag{4.1}$$

El algoritmo requiere de una estimación inicial para el vector $\mathbf{a}$. En cada iteración sus elementos son reemplazados por un valor δ.

$$f(m_j, \mathbf{a} + \delta) \approx f(m_j, \mathbf{a}) + J_j \delta \tag{4.2}$$

con:

$$J_j = \frac{\partial f(m_j, \mathbf{a})}{\partial \mathbf{a}} \tag{4.3}$$

$$E(\mathbf{a}+\delta) \approx \sum_{i=1}^{n}(m_i - f(d_i,\mathbf{a}) - J_i\delta)^2 \tag{4.4}$$

Empleando notación matricial:

$$E(\mathbf{a}+\delta) \approx ||m_i - f(d_i,\mathbf{a}) - \mathbf{J}\delta||^2 \tag{4.5}$$

La matriz **J** es conocida como el $Jacobiano$[2], donde cada renglón J_i, toma la derivada de la ecuación (4.5) respecto a δ. Igualando a cero resulta:

$$(\mathbf{J}^T\mathbf{J})\delta = \mathbf{J}^T[\mathbf{d} - \mathbf{f}(\mathbf{a})] \tag{4.6}$$

Levenberg hizo una observación y agregó al resultado un factor de amortiguamiento λ:

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\delta = \mathbf{J}^T[\mathbf{d} - \mathbf{f}(\mathbf{a})] \tag{4.7}$$

El valor de λ se actualiza en cada iteración. Si la reducción de E es rápida, pequeños valores pueden ser usados y el algoritmo funciona de manera similar al método de Gauss-Newton, pero si la reducción de E es mínima, λ puede ser incrementada en cada iteración para que el algoritmo trabaje de forma similar a un método de gradiente. Una desventaja del algoritmo LM es que para valores grandes de λ el algoritmo no funciona correctamente, por lo cual Marquardt propuso una mejora al algoritmo de Levenberg reemplazando la matriz de identidad por el valor de diag($\mathbf{J}^T\mathbf{J}$).

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{diag}(\mathbf{J^T}\mathbf{J}))\delta = \mathbf{J}^T[\mathbf{d} - \mathbf{f}(\mathbf{a})] \tag{4.8}$$

[2]La matriz jacobiana es una matriz formada por las derivadas parciales de primer orden de una función

4.3.1 Complejidad algoritmo del Levenberg-Marquardt

En los trabajos publicados en [61, 62], se presenta la estimación de un límite superior de iteraciones para que el algoritmo de Levenberg-Marquardt converja a una solución.

Dada una función de múltiples variables:

$$\phi(x) = \frac{1}{2}||F(x)||^2$$

donde: $F : \Re^n \rightarrow \Re^m$ es diferenciable.

La complejidad del algoritmo Levenberg-Marquardt, considerando el límite superior de iteraciones necesarias para obtener una solución tal que:

$$||\nabla\phi(x^k)|| = ||J(x^k)^T F(x^k)|| \leq \epsilon$$

donde: x^k representa el punto x en la iteración k y ∇ el vector gradiente de la función ϕ, está definida por el teorema descrito en [61, 62]. Un número total de iteraciones es:

$$K_{total} = \mathcal{O}(\epsilon^{-2})$$

En la figura 4.2 se gráfica el comportamiento de la complejidad del algoritmo de Levenberg-Marquardt. La complejidad del algoritmo reportada en estos trabajos sólo son un límite superior del total de iteraciones, aunque el algoritmo bien podría terminar en una cantidad menor de iteraciones [61, 62].

4.4 Metodología propuesta para registro de nubes de puntos

En esta sección se presenta la descripción del problema que se aborda como parte de este trabajo y la metodología propuesta para resolverlo.

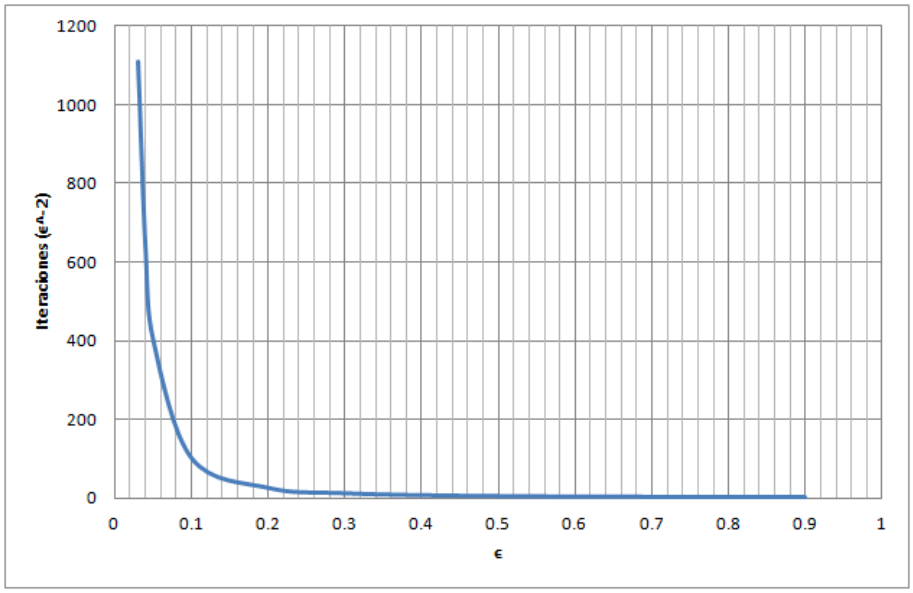


Figura 4.2: Gráfica del comportamiento de la complejidad del algoritmo de Levenberg-Marquardt

4.4.1 Propuesta para registrar nubes de puntos con descriptores de superficie

Por notación se llamará $modelo$ a los conjuntos de datos denotados por: $\{\mathbf{m}_i\}_{i=1}^{n}$ y $\{\mathbf{s}_j\}_{j=1}^{k}$, donde $\mathbf{m}_i$ $(i = 1, 2, \cdots, n)$ son los elementos de un conjunto de puntos 3D y $\mathbf{s}_j$ $(j = 1, 2, \cdots, k)$ son los elementos de sus descriptores de superficie, respectivamente. Se llamará $datos$ a los conjuntos de datos denotados por: $\{\mathbf{d}_i\}_{i=1}^{p}$ y $\{\mathbf{v}_j\}_{j=1}^{q}$, donde $\mathbf{d}_i$ $(i = 1, 2, \cdots, p)$ son los elementos de un conjunto de puntos 3D y $\mathbf{v}_i$ $(i = 1, 2, \cdots, q)$ son los elementos de sus descriptores de superficie, respectivamente. En el presente trabajo se aborda el problema de buscar una transformación rígida T, que aplicada a los puntos 3D y a los descriptores de superficie de los elementos en $datos$, se ajuste mejor a los elementos del $modelo$.

Una transformación rígida involucra una matriz de rotación $R \in \Re^{3\times3}$ y un vector de traslación $\mathbf{t} \in \Re^3$. La matriz de rotación puede ser descrita por el producto de tres matrices, que describen la rotación en cada uno de los principales tres ejes (xyz), con

sus correspondientes ángulos de Euler α, β y γ. Se define esta rotación como: $R = R_z(\gamma)R_y(\beta)R_z(\alpha)$, donde R_y y R_z son definidas como:

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \operatorname{sen}\beta \\ 0 & 1 & 0 \\ -\operatorname{sen}\beta & 0 & \cos\beta \end{bmatrix}$$

y

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\operatorname{sen}\gamma & 0 \\ \operatorname{sen}\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Juntando los parámetros de la matriz de rotación R y los del vector de traslación **t**=$[t_x, t_y, t_z]$ en el vector **a**=$[\alpha, \beta, \gamma, t_x, t_y, t_z]$, la transformación T aplicada a un punto **x** se define como:

$$T(\mathbf{a}; \mathbf{x}) = R\mathbf{x} + \mathbf{t}$$

Para un vector $\mathbf{v}$ definimos su transformación como:

$$T(\mathbf{a}; \mathbf{v}) = R\mathbf{v}$$

Para resolver el problema del registro primero es necesario conocer algunas correspondencias entre los conjuntos de puntos de la figura $modelo$ y $datos$, y ya con las correspondencias estimadas se evalúa el error que hay entre ambos datos. Para evitar una búsqueda exhaustiva en los conjuntos de datos, se empleó la estructura de datos descrita en la sección 4.2 (o sea, los árboles-kd). Cada punto del conjunto de prueba $datos$ busca una y sólo una correspondencia en el conjunto $modelo$ cumpliendo con la restricción de ser menor a una constante τ $(0 < \tau < 1)$.

Aunque; el problema del registro puede resolverse solamente empleando información

de los puntos que describen un objeto, en este trabajo se enfocó en usar descriptores de superficie tales como: planos, esferas, cilindros y conos, que nos permitan hacer el registro en una forma más eficiente. Los descriptores de superficie bien representan todos los puntos de un área del objeto que fueron ajustados previamente. Los descriptores de superficie empleados son los que se propusieron en el capítulo 3 de este trabajo.

En este trabajo se utilizó el producto punto o producto interno de dos vectores para evaluar la orientación entre vectores, definido por:

$$\mathbf{v} \cdot \mathbf{u} = \|\mathbf{v}\| \, \|\mathbf{u}\| \cos\theta \tag{4.9}$$

De acuerdo a la ecuación (4.9) dos vectores son paralelos o tienen la misma orientación si el ángulo θ entre ellos es 0 radianes. Para usar este hecho, se agregó esta métrica a nuestra función objetivo, para poder estimar el error entre vectores que describen superficies. Por lo cual, nuestra función objetivo queda definida por:

$$E(\mathbf{a}) = \sum_{i=1}^{n} (\mathbf{m}_i - T(\mathbf{a}, \mathbf{d}_i)) + w \sum_{j=1}^{k} \operatorname{arc\,cos}\left(\frac{\mathbf{s}_j \cdot T(\mathbf{a}, \mathbf{v}_j)}{|\mathbf{s}_j||T(\mathbf{a}, \mathbf{v}_j)|}\right) \tag{4.10}$$

y la optimización de los parámetros del vector **a** es obtenida mediante la minimización de:

$$\hat{\mathbf{a}} = \operatorname{argmin} E(\mathbf{a}) \tag{4.11}$$

El valor de la constante w en la ecuación (4.10), la cual nos permite dar mayor peso al error del ajuste entre superficies, fue puesta experimentalmente a 25 y este valor fue considerado igual en la etapa de pruebas.

El algoritmo de Levenberg-Marquardt requiere estimar una matriz de derivadas (matriz Jacobiana **J**) respecto a cada uno de los parámetros de **a**. Este problema puede ser resuelto utilizando el método de diferencias finitas [64, 42, 8]. Este método es útil para funciones o datos que no tienen una derivada exacta. Investigaciones hechas en el contexto de métodos numéricos muestran la utilidad de este método para resolver este problema [64, 42, 8].

En un paso previo a la optimización de la función objetivo se requieren calcular las correspondencias entre los puntos del $modelo$ y los $datos$. Para esto se definió la función ϕ, la cual busca el punto más cercano entre los conjuntos de datos. En el algoritmo 3 se describe la forma de operar de la función ϕ.

Algoritmo 3 Calcular puntos de correspondencia (ϕ)

Entrada: Nube de puntos $modelo$, $datos$, tolerancia τ

1: $kdt = kdtree(modelo)$
2: $corr = kdt.query(datos)$
3: **para** i=1 to $card(corr)$ **hacer**
4: **si** $corr_i < \tau$ **entonces**
5: $\phi \leftarrow corr_i$
6: **fin si**
7: **fin para**
8: $removerDuplicados(\phi)$
9: **regresa** ϕ

Para hacer la correspondencia entre vectores que describen superficies, se desarrolló la función ϕ_2, la cual evalúa el mínimo ángulo entre vectores usando la ecuación (4.9) y asigna la correspondencia entre cada vector. El algoritmo 4 describe la operación de la función ϕ_2.

Algoritmo 4 Calcular correspondencias entre vectores (ϕ_2)

Entrada: Conjunto de vectores **s**, **v**, tolerancia τ_2

1: **para** i=1 to $card(\mathbf{s})$ **hacer**
2: **para** j=1 to $card(\mathbf{v})$ **hacer**
3: **si** $corr = evaluarAngulo(\mathbf{s}_i, \mathbf{v}_j) < \tau_2$ **entonces**
4: $\phi_2 \leftarrow corr$
5: **fin si**
6: **fin para**
7: **fin para**
8: $\phi_2 = removerDuplicados(\phi_2)$
9: **regresa** ϕ_2

Para inicializar el algoritmo, primero se calcula el centro de masa de los puntos del conjunto $datos$ y $modelo$, y se traslada el conjunto $datos$ al centro de masa del conjunto $modelo$ y después se aplican las funciones ϕ y ϕ_2 para buscar la mejor correspondencia

entre puntos y vectores, respectivamente. Puntos que no tienen una correspondencia son removidos del proceso de optimización. El vector de parámetros **a** se inicializa con: $\alpha = \beta = \gamma = 0$ y **t** = centro de masa del conjunto de puntos $modelo$.

También se implementó una función E que estima el error en cada iteración. La función toma como entrada los puntos con sus correspondencias, calculados por la función ϕ, y los vectores con sus correspondencias, calculados por la función ϕ_2. La función E construye la matriz de rotación R tomando los ángulos de rotación en los principales ejes (xyz) y construye también el vector de traslación **t**; posteriormente se aplica la transformación a los puntos y vectores del conjunto $datos$ y se calcula el error obtenido mediante esta transformación. La función devuelve un vector de errores por cada punto de la siguiente forma:

$$e_i = [\Delta x_i, \Delta y_i, \Delta z_i]^T$$

y por cada correspondencias entre vectores regresa un vector de la siguiente forma:

$$e_j = [25\theta_j]$$

El error final es la concatenación de los vectores: $\mathbf{e}_i$ y $\mathbf{e}_j$.

El algoritmo 5 describe los pasos que se realizan para resolver el registro de nubes de puntos empleando vectores como descriptores de superficie. La implementación de la propuesta fue hecha en el leguaje de programación Python versión 2.7.6. Además, se utilizó el paquete de optimización LMFIT [38], el cual tiene una implementación del algoritmo Levenberg-Marquardt. Se empleó una computadora con sistema operativo Windows 7 con procesador intel i5 de 64 bits, velocidad de 2.5GHz y 4GB de memoria RAM.

Algoritmo 5 Registro con superficies

Entrada: $\{\mathbf{m}\}$, $\{\mathbf{s}\}$, $\{\mathbf{d}\}$ y $\{\mathbf{v}\}$

1: Buscar correspondencias ϕ y ϕ_2
2: inicializa $\alpha = \beta = \gamma = 0$
3: inicializa $\mathbf{t} = centroide(\mathbf{m})$
4: inicializa $\hat{\mathbf{a}} = \{\alpha, \beta, \gamma, \mathbf{t}\}$
5: **repetir**
6: $\quad \hat{\mathbf{a}} = minimizar(\mathbf{m}, \mathbf{d}, \mathbf{s}, \mathbf{v}, \phi, \phi_2)$
7: $\quad \mathbf{d}' = T(\hat{\mathbf{a}}; \mathbf{d})$
8: $\quad \mathbf{e} = [\Delta x_i, \Delta y_i, \Delta z_i]^T$
9: $\quad \mathbf{s}' = T(\hat{\mathbf{a}}; \mathbf{s})$
10: $\quad \mathbf{e} \leftarrow \Delta \mathbf{s}'$
11: $\quad E = \mathbf{e}^T \cdot \mathbf{e}$
12: **hasta** Mínimo local
13: **regresa** $\hat{\mathbf{a}}$

4.5 Experimentación y resultados

Para evaluar la metodología propuesta se realizaron cuatro diferentes experimentos. La primera prueba se realizó empleando los datos de la figura 4.3. El cubo se seleccionó como el conjunto $modelo$ y el tetraedro se seleccionó como el conjunto de $datos$ a registrar. Se generó un conjunto de datos sintéticos para producir un cubo unitario, el cual es representado por 8 puntos 3D (cada punto es un vértice del cubo) y 3 vectores que definen la orientación de tres de sus 6 caras. Igualmente, se generaron los 4 puntos para el tetraedro y los vectores en 3 de sus 4 caras.

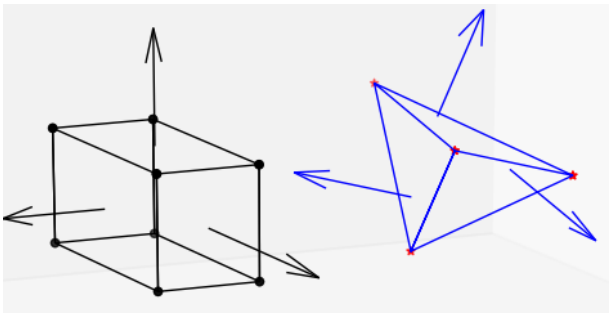


Figura 4.3: Ejemplo 1. Cubo unitario utilizado como $modelo$ y el tetraedro a ser registrado. En ambas figuras se muestran los vectores utilizados en el proceso de registro

La segunda prueba consistió en un paralelepipedo recto utilizado como $modelo$ y un cubo unitario para ser registrado. Los datos de los vértices y los vectores de las figuras geométricas fueron generados sintéticamente, es decir los 8 vértices por cada figura y sus respectivos vectores en 3 de sus respectivas 6 caras. Las gráficas de las piezas se muestran en la figura 4.4.

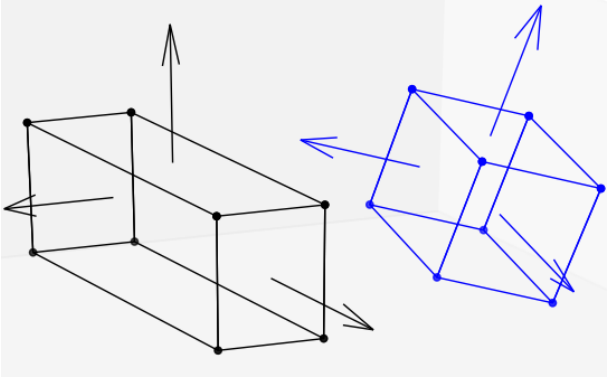


Figura 4.4: Ejemplo 2. Paralelepípedo recto usado como $modelo$. Cubo unitario a registrar. En ambas figuras se muestran los puntos y vectores empleados en el algoritmo de registro propuesto en este trabajo

En el proceso de registro no se emplearon todas las correspondencias de puntos, solamente se seleccionó aleatoriamente una correspondencia de su conjunto. Sin embargo, sí se emplearon todos los vectores que describen la orientación de los objetos. Los vectores nos describen la orientación de las caras de la figuras, pero para obtener la posición final del objeto a registrar es necesario especificar al menos un punto de control para logar el posicionamiento correcto.

Los resultados de 31 ejecuciones para las pruebas 1 y 2 se muestran en las tablas 4.2 y 4.3 respectivamente. En cada ejecución se hizo una rotación y una traslación aleatoria aplicada a la figura a ser registrada. Las figuras 4.5 y 4.6 muestran el registro obtenido después de aplicar la metodología propuesta.

La tercera prueba se realizó utilizando datos reales generados por el escáner 3D desarrollado en este trabajo. El objeto del mundo real es un adaptador USB, el cual fue seleccionado por estar construido con primitivas geométricas como planos y cilindros. La

Tabla 4.2: Resultados estadísticos para 31 ejecuciones para los datos de la figura 4.3

	Iteraciones	Tiempo [seg.]	Error
Promedio	31.7	0.0085	4.24×10^{-8}
Max.	36	0.0099	1.31×10^{-7}
Min.	29	0.0055	3.06×10^{-9}
Desviación estándar	3.19	0.0012	5.24×10^{-8}

Tabla 4.3: Resultados estadísticos para 31 ejecuciones para los datos de la figura 4.4

	Iteraciones	Tiempo [seg.]	Error
Promedio	55.85	0.0194	1.17×10^{-3}
Max.	67	0.0274	1.72×10^{-3}
Min.	36	0.0119	1.88×10^{-8}
Desviación estándar	9.7	0.0057	5.0×10^{-4}

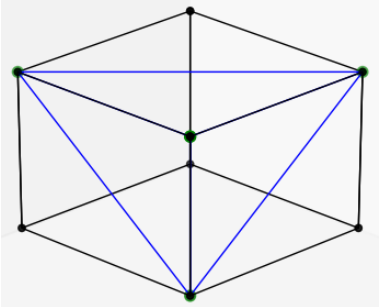


Figura 4.5: Resultado final de registrar los objetos de la figura 4.3

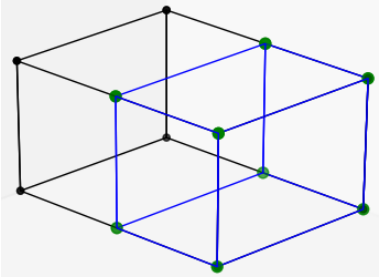


Figura 4.6: Resultado final de registrar los objetos de la figura 4.4

forma de estimar los vectores normales de los planos y los vectores directores de los cilindros es el descrito en el capítulo 3. El objeto $modelo$ consiste de una nube de puntos tomada con el escáner con un total de 3,664 puntos 3D. El objeto a registrar es una nube de puntos generada por el escáner en un segundo proceso de escaneo del conector USB.

El archivo contiene un total de 5,461 puntos 3D.

Los resultados del desempeño de la metodología para registrar estos conjuntos de datos se muestran en la tabla 4.4. Los resultados muestran un promedio de ejecución de 0.1117 segundos.

Como ejercicio adicional se realizó el registro de las mismas nubes de puntos de esta prueba, pero empleando sólo información de los puntos, después de 15 ejecuciones, el algoritmo tardó en promedio: 2.01 segundos, lo cual representa una mejora en el tiempo de ejecución de 1,799 %.

Tabla 4.4: Resultados estadísticos para 31 ejecuciones correspondientes a los datos de la figura 4.9

	Iteraciones	Tiempo [seg.]	Error
Promedio	496.81	0.111717276	3.28×10^{-2}
Max.	500	0.147331613	3.31×10^{-2}
Min.	492	0.103919749	3.23×10^{-2}
Desviación estándar	2.891995222	0.013049612	2.87×10^{-4}

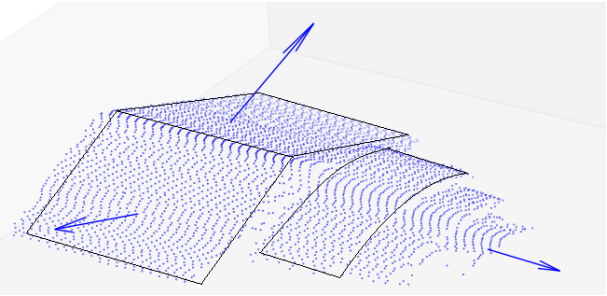


Figura 4.7: Nube de puntos $modelo$. En esta imagen se muestra un adaptador USB con dos de sus caras ajustadas a dos planos. También se muestran tres cilindros que son descritos por sus vectores directores

El cuarto ejercicio que se realizó fue hecho con un objeto escaneado en dos partes. El registro fue realizado sin información alguna de la nube de puntos, sino sólo con la información de los descriptores de superficie. En la escena de la figura 4.11 se pueden considerar dos planos, una esfera, un cilindro y un cono por cada toma del escáner. El modelo de color azul contiene 16,048 puntos 3D y la nube de puntos a registrar en color

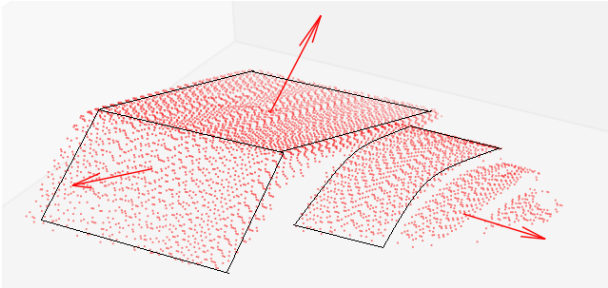


Figura 4.8: Nube de puntos $datos$. En esta imagen se muestra un adaptador USB con dos de sus caras ajustadas a dos planos. También se muestran tres cilindros que son descritos por sus vectores directores

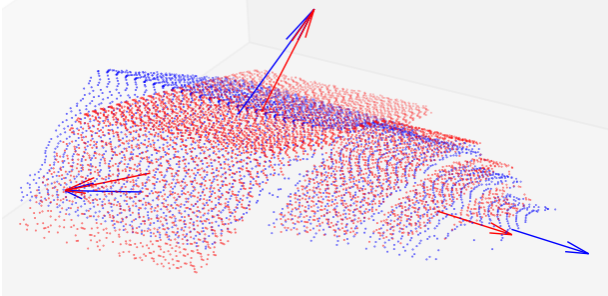


Figura 4.9: Inicialización de las nubes de puntos $modelo$ y $datos$ antes del proceso de registro. El registro básicamente es hecho con información de los vectores que describen los planos y los cilindros en las nubes de puntos

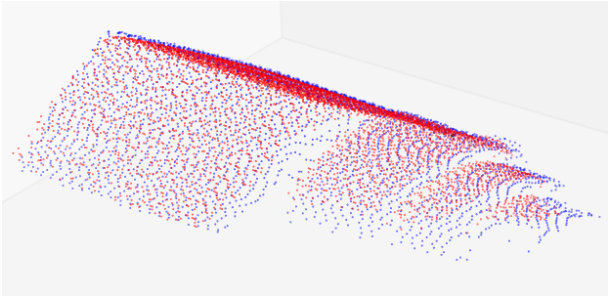


Figura 4.10: Registro final obtenido después de aplicar la metodología propuesta de la figura 4.9

rojo contiene 12,169 puntos 3D.

En la tabla 4.5 se muestra el promedio de iteraciones, tiempo de ejecución y error después de 31 ejecuciones usando la metodología propuesta.

Los resultados obtenidos al implementar la metodología para el registro de nubes de

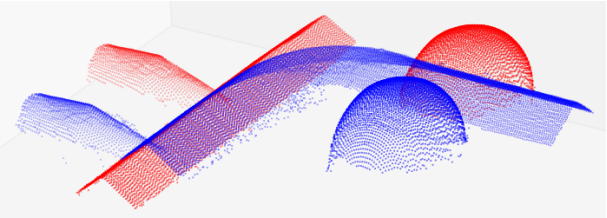


Figura 4.11: Inicialización de los objetos a registrar. El registro de estas figuras fue hecho usando solamente la información de los descriptores de superficie: dos planos, un cilindro, un cono y una esfera. Puntos en color azul representan el conjunto $modelo$, puntos en color rojo representan el conjunto $datos$

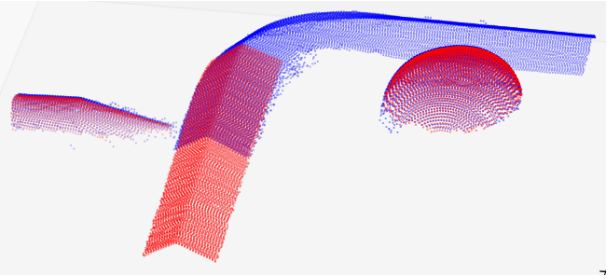


Figura 4.12: Resultado final del registro de los datos de la figura 4.11 empleando la metodología propuesta en este trabajo. Puntos en color azul representan el conjunto $modelo$, puntos en color rojo representan el conjunto $datos$

Tabla 4.5: Resultados estadísticos para 31 ejecuciones para los datos de la figura 4.12

Iteraciones	Tiempo [seg.]	Error
462	0.115297727	3.85×10^{-2}

puntos mediante descriptores de superficie, nos permiten notar una mejora notable en tiempo de ejecución que usar solamente puntos.

5

Conclusiones

La investigación en áreas como la visión por computadora y sus áreas afines es extensa y requiere del desarrollo de instrumentos para la adquisición de información del mundo real, además de requerir de instrumentos como los digitalizadores 3D. También se necesitan metodologías que permitan extraer características contenidas en los datos, para así poder generar representaciones computacionales que sean más eficientes y fáciles de procesar.

En este trabajo se presentó el desarrollo y construcción de un escáner 3D compuesto por una mesa de pruebas pasiva y cuatro módulos principales: módulo mecánico, módulo de sensor, módulo de software y módulo de hardware. El escáner construido en este trabajo permite hacer la digitalización de objetos del mundo real, es decir, permite generar la nube de puntos que describe la superficie de tales objetos.

Se realizaron dos pruebas para la caracterización del error en las mediciones hechas por el escáner en condiciones normales de operación:

- La primera prueba se hizo con un objeto de superficie plana. Los datos recolectados en la nube de puntos se utilizaron para ajustar los parámetros de la ecuación de un plano, mediante una técnica de descomposición de una matriz cuadrada en sus

eigenvalores y eigenvectores. La comparación de los valores observados por el escáner contra el modelo matemático generado por el ajuste, nos arrojó como resultado un error promedio de $\pm 3.67\mu$m en la dirección z (ver sección 2.3.1, en la página 26).

- El segundo ejercicio se realizó con un objeto de superficie esférica, los datos recolectados en su nube de puntos correspondiente fueron utilizados para ajustar los parámetros de la ecuación de una esfera, mediante un método de minimización del error cuadrático medio. La comparación de los puntos generados por el escáner contra el modelo estimado nos dió un error promedio de $\pm 1.23\mu$m en la dirección z (ver sección 2.3.2, en la página 30).

- La precisión en el eje x esta dado por el módulo lineal, de acuerdo a las hojas técnicas del fabricante es de $\pm 5\mu$m. La precisión en dirección horizontal esta dada por el sensor, de acuerdo a sus hojas técnicas es de 11 μm .

Se utilizó el paradigma MAC para detectar primitivas geométricas contenidas en las nubes de puntos, mediante el ajuste de los parámetros de las ecuaciones de: planos, esferas, cilindros y conos. Este proceso se utilizó como método de segmentación o extracción de primitivas de las nubes de puntos. Las primitivas que fueron extraídas de las nubes de puntos, nos permitieron hacer una reconstrucción del objeto o de la escena 3D mediante la técnica de modelado llamada: Geometría constructiva de sólidos (GCS, o por su nombre en inglés *Constructive Solid Geometry* (GSD)), la cual permite, mediante una serie de modelos, reconstruir objetos o escenas del mundo real de una forma más precisa y eficiente que empleando una nube de puntos.

El desempeño de la metodología propuesta para extraer primitivas geométricas, fue mejorado al disminuir la cantidad de iteraciones necesarias para converger, al hacer uso del conocimiento de que los datos están ordenados y nos son aleatorios. Entonces la muestra de concenso se busca a cada dos líneas de escaneo en vez de en todos los puntos escaneados. Esto dió como resultado una mejora de 737 % con respecto al número de iteraciones requeridas por un muestreo uniforme.

También se presentó en este trabajo, una metodología para el registro de nubes de puntos empleando descriptores de superficie. Los descriptores de superficie utilizados fueron vectores que describen: planos, cilindros y esferas. Adicionalmente, se agregó un punto 3D como descriptor del centro de una esfera. La metodología propuesta toma algunos de los pasos del algoritmo PMCI y mediante la utilización de un optimizador de uso general, como el propuesto por Levenberg-Marquardt, se pudo alcanzar el registro final. Al hacer uso de esta metodología se mostró que es posible hacer el registro de nubes de puntos, empleando sólo los descriptores de superficie obtenidos a través de un proceso de segmentación. La idea de esta metodología es que una superficie puede representar un subconjunto de puntos 3D del total del conjunto de puntos de la nube de puntos, permitiendo así mejorar el tiempo de procesamiento del registro. De los datos obtenidos en este trabajo se puede notar que hay una mejora en tiempo de aproximadamente 1,799 % contra la misma metodología pero usando sólo puntos 3D.

Las conclusiónes finales son las siguientes:

- Un escáner 3D puede ser construido con diferentes tipos de tecnologías. Una que presenta buena precisión es la tecnología láser con un método de medición por triangulación.

- Un escáner puede ser construido de forma modular para producir un sistema de escaneo escalable y reconfigurable de acuerdo a las necesidades de la aplicación.

- Para hacer la reconstrucción de un objeto o escena del mundo real, se puede primero hacer una segmentación de los objetos geométricos contenidos en sus nubes de puntos, para posteriormente hacer una representación mediante el uso de modelos computacionales.

- Los objetos del mundo real, más grandes de lo que puede escanear el sensor en una sola pasada, comúnmente requieren ser reconstruidos a partir de dos o más nubes de puntos. Para lograr esto de forma eficiente se pueden emplear metodologías que

permitan resolver este problema como el uso de minimización del error de orientación de las superficies que describen tales objetos.

5.1 Trabajo futuro

Algunas posibles líneas de trabajo futuro son las siguientes:

1. Construir una interfaz gráfica para incorporar todos los algoritmos diseñados en esta tesis. Esto nos permitiría escanear objetos de forma semiautomática, con la ayuda de un usuario.

2. Realizar el proceso del punto anterior de forma automática.

3. Para el paso anterior tal vez se requeriría añadir cierta inteligencia al escáner para reconocer automáticamente el objeto y poder digitalizar todas sus partes.

4. Añadir un descriptor de superficie general, tal vez parches de splines, que puedan describir cualquier superficie curva. Encontrar el número mínimo de estos parches es un problema difícil [47, 71].

Bibliografía

[1] Faro focus 3D escanner products, http://www.faro.com /products/3d-surveying/laser-scanner-faro-focus-3d/overview, última visita mayo- 2016.

[2] Laser design 3D escanner products, http://www.laserdesign.com /products/category/highest-accuracy-3d-scanners/, última visita mayo- 2016.

[3] Next engine 3D escanner products, http://www.nextengine.com/products, última visita mayo- 2016.

[4] *Computational Geometry: Algorithms and Applications*, chapter Orthogonal Range Searching, pages 95–120. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[5] J Dan A. M. Brito J, A. Duarte Doria N. An adaptive learning approach for 3-d surface reconstruction from point clouds. In *IEEE Trans. On Neural Networks*, pages 1–11, 2006.

[6] P. Novák A. Miks, J. Novák. Linear method for evaluation of radius of spherical surface from discrete set of data points. *Optik - International Journal for Light and Electron Optics*, 124(22):5473–5477, 2013.

[7] R.B. Fisher A. W. Fitzgibbon, M. Pilu. Direct least square fitting of ellipses. *IEEE Trans. of Pattern Analysis and Machine Intelligence*, 21(5):476–480, 1999.

[8] M. H. Chaudhry. *Open-Channel Flow*. Springer, London, 1 edition, 2008.

[9] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 905–914, New York, NY, USA, 2004. ACM.

[10] Brian Curless. From range scans to 3d models. *ACM SIGGRAPH Computer Graphics*, 33(4):88–41, 2000.

[11] S. Nasuto D. Myatt, P. Torr. Napsac: High noise, high dimensional robust estimation its in the bag. In *BMVC, Computer Vision Tools*, 2002.

[12] L.G. de la Fraga. A very fast procedure to calculate the smallest singular value. In *Eighth International Conference on Advances in Pattern Recognition (ICAPR)*, pages 1–4, Kolkata, India, 4–7 Jan. 2015.

[13] Manual de usuario de Sensor Inteligente de la Serie ZG. http://www.omronkft.hu/pdf en/zg_user_man.pdf, última visita mayo- 2016.

[14] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. In *Communications of the ACM*, pages 11–15, 1972.

[15] J. Salvi. et al. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5):578 – 596, 2007.

[16] T. Wu et al. Automatic 3d point clouds registration method. In *Proceedings of SPIE, 7855*, volume 26, 2010.

[17] Y.F. Wu et al. A new method for registration of 3d point sets with low overlapping ratios. In *Conference on Computer Aided Tolerancing*, volume 27, pages 202 – 206, 2015.

[18] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Communications of the ACM*, pages 381–395, 1983.

[19] A. W. Fitzgibbon. Robust registration of 2d and 3d points set. *Image and Vision Computing*, 21(13–14):1145–1153, 2003.

[20] M.S. Lee G. Medioni and C.K. Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier Science, 1 edition, 2000.

[21] M. Muñoz M. Fernández-Guasti G. Muñoz, C. García. Ultra stable holographic table: analysis of vibrations. In *In XXIII Congreso de óptica. Puebla, Pue. México*, 2010.

[22] F. Docchio G. Sansoni, M. Trebeschi. State-of-the-art and applications of 3d imaging sensors in industry, cultural heritage, medicine, and criminal investigation. 9(1):568–601, 2009.

[23] G. Sithole T. Rabbani G. Vosselman, B.G.H. Gorte. Recognising structure in laser scanner point clouds. In *46th. Inter. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, pages 33–38, 2004.

[24] G. H. Golub and C. F. Van Loan. *Matrix computation*. The Johns Hopkings University Press, Baltimore, USA, fourth edition, 2013.

[25] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[26] G. R. Gordon. The laser, light amplification by stimulated emission of radiation, the ann arbor conference on optical pumping. In *In Franken, P.A. and Sands, R.H. (Eds.).*, page 128, 1959.

[27] M. Greenspan and G. Godin. A nearest neighbor method for efficient icp. In *Proceedings. Third International Conference on 3-D Digital Imaging and Modeling*, pages 161 – 168, 2001.

[28] A. Gruen and D. Akca. Least squares 3d surface and curve matching. *Journal of Photogrammetry and Remote Sensing*, 59(3):151 – 174, 2005.

[29] Y. Sun H. Cheng, X. Jiang and J. Wang. Color image segmentation: Advances and prospects. 34(12):2259–2281, 2001.

[30] R. I. Hartley. Euclidian reconstruction from uncalibrated views. In *Applications of Invariance in Computer Vision*, pages 237–256, 1993.

[31] D. Borrmann J. Elseberg and A. Nuchter. Efficient processing of large 3d point clouds. In *XXIII International Symposium on Information, Communication and Automation Technologies (ICAT)*, pages 1–7, Sarajevo, 2011.

[32] Yamashita N. Fukushima M. Kanzow, C. Levenberg-marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints. *Appl. Math. Comput, 173*, pages 321–343, 2005.

[33] Módulos lineales de alta precisión rexroth. https://www.boschrexroth.com /en/xc/products/product-groups/linear-motion-technology/linear-motion-systems/precision-modules/index, última visita mayo- 2016.

[34] H. Pottmann T. Steiner J. Wallner M. Hofer, B. Odehnal. 3d shape recognition and reconstruction based on line element geometry. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) (Volume:2)*, pages 1532 – 1538, 2005.

[35] C. Bradley M. J. Milroy, D. J. Weir and G. W. Vickers. Reverse engineering employing a 3d laser scanner: A case study. 12:111–121, 1996.

[36] Tang J. Ma, C. The quadratic convergence of a smoothing levenberg-marquardt method for nonlinear complementarity problem. *Appl. Math. Comput, 197*, pages 566–581, 2008.

[37] Tang J. Chen X. Ma, C. A globally convergent levenberg-marquardt method for solving nonlinear complementarity problem. *Appl. Math. Comput, 192*, pages 370 – 381, 2007.

[38] Non-Linear Least-Square Minimization and Curve-Fitting for Python. https://lmfit.github.io/lmfit-py/, última visita mayo- 2016.

[39] J. J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory, numerical analysis. In *Watson, Lecture Notes in Mathematics, Springer Verlag*, pages 105–116, 1977.

[40] P. Mukhopadhyaya and B. B. Chaudhuria. A survey of hough transform. *Pattern Recognition*, 48(3):993–1010, 2015.

[41] Gelfand N. and L. Guibas. Shape segmentation using local slippage analysis. In *Proc. Symp. Geom. Processing*, 2004.

[42] P. Olver. *Introduction to Partial Differential Equations*. Undergraduate Texts in Mathematics. Springer, 2014.

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

[44] A. Rosenfeld P. Meer, D. Mintz and D. Y. Kim. Robust regression methods for computer vision: A review. *International Journal of Computer Vision*, 6(1):9–70, 1991.

[45] LAPACK Linear Algebra PACKage. http://www.netlib.org/lapack/, última visita mayo-2016.

[46] Elvezio M. Ronchetti Peter J. Huber. *Robust Statistics*. Wiley, second edition, 2009.

[47] J. Peters and J. Fan. On the complexity of smooth spline surfaces from quad meshes. *Computer Aided Geometric Design*, 27(1):96 – 105, 2010.

[48] V. Pratt. Direct least-squares fitting of algebraic surfaces. *ACM SIGGRAPH Computer Graphics*, 21(4):141–152, 1987.

[49] Brown R.A. Building a balanced k-d tree in o(kn log n) time. 4(1):50–68, 2015.

[50] Richard E. Woods Rafael C. Gonzalez. *Digital Image Processing*. Publishing House of Electronics Industry, Beijing, 2007.

[51] P. J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(38):871–880, 1984.

[52] S. Rusinkiewicz and M Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *In Siggraph*, pages 343–352, 2000.

[53] M. Bennamoun S. A. Ali Shah and F. Boussaid. Performance evaluation of 3d local surface descriptors for low and high resolution range image registration. In *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7, Wollongong, NSW, Nov. 2014.

[54] H. D. Enge S. J. Buckley, J. A. Howell and H. D. Kurz. Terrestrial laser scanning in geology: data acquisition, processing and accuracy considerations. 12:625–638, 2008.

[55] M. Levoy S. Rusinkiewicz. Efficient variants of the icp algorithm. In *3 Int. Conference on 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[56] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007.

[57] Copyright 1999 2010 Thorlabs Sources of vibration, theory of tabletop vibration. http://www.thorlabs.com/tutorials/tables.cfm, última visita mayo-2016.

[58] Hojas técnicas IC L298. https://www.sparkfun.com /datasheets/robotics/l298_h_bridge.pdf, hojas técnicas ic l298, última visita mayo-2016.

[59] Hojas técnicas PIC–18F4550. http://ww1.microchip.com /downloads/en/devicedoc/39632e.pdf, última visita mayo- 2016.

[60] P. H. S Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24(3):271–300, 1997.

[61] Kenji Ueda and Nobuo Yamashita. On a global complexity bound of the levenberg-marquardt method. *Journal of Optimization Theory and Applications*, 147(3):443–453, 2010.

[62] Kenji Ueda and Nobuo Yamashita. Global complexity bound analysis of the levenberg–marquardt method for nonsmooth equations and its application to the nonlinear complementarity problem. *Journal of Optimization Theory and Applications*, 152(2):450–467, 2012.

[63] G. W. Vickers V. H. Chan, C. Bradley. A multi-sensor approach to automating co-ordinate measuring machine-based reverse engineering. *Computers in Industry*, 44(2):105–115, 2001.

[64] W. T. Vetterling W. H. Press, S. A. Teukolsky and B. P. Flannery. *Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[65] Havran V Wald I. On building fast kd-trees for ray tracing, and on doing that in o(n log n). In *Proceedings of the IEEE Symposium on Interactive Ray Tracing*, pages 61–69, 2006.

[66] Delorme FE Engin S. Wang Y, Feng HY. Evaluation of normal vector estimation methods for scanned point clouds. In *Proceedings of the CIRP 1st international conference on virtual machining process technology*, page paper No. 12, 2012.

[67] CH. Menq WL. Chen. Integrated laser/cmm system for the dimensional inspection of objects made on soft material. *The Int. Journal of Adv. Manu. Tech.*, 10(1):36–45, 1995.

[68] C. Wöhler. *3D Computer Vision: Efficient Methods and Applications*. Springer (X.media.publishing), London, 1 edition, 2013.

[69] J. Wu and L. Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum (Proc. Eurographics*, pages 277–284, 2005.

[70] O. Wulf and B. Wagner. Fast 3d scanning for laser measurement systems. In *Int. Conf. on Control systems eng.*, pages 312–317, 2003.

[71] P. Wu X. Shi, T. Wang and F. Liu. Reconstruction of convergent smooth b-spline surfaces. *Computer Aided Geometric Design*, 21(9):893 – 913, 2004.

[72] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.

[73] Z. Zhang. Parameter estimation technique: A tutorial with application to conic fitting. *Image and Vision Computing*, 15(1):59–76, 1997.