Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Sistema de escaneo 3D integrado y de bajo costo usando luz estructurada y súper-resolución

Tesis de Licenciatura en Ciencias de la Computación

Christian Cuneo

Director: Francisco Gómez Fernández

Buenos Aires, 2020

SISTEMA DE ESCANEO 3D INTEGRADO Y DE BAJO COSTO USANDO LUZ ESTRUCTURADA Y SÚPER-RESOLUCIÓN

El escaneo 3D es utilizado en diversas áreas de investigación tanto científicas como de ingeniería. En la actualidad los escáneres 3D de bajo costo orientados a la investigación requieren un conocimiento técnico complejo previo para su configuración y uso, mientras que los orientados al desarrollo de juegos y aplicaciones no sólo carecen de garantías claras de precisión y exactitud, sino que al ser diseñados para desarrollo, no tienen una interfaz de usuario. Por otro lado, los escáneres 3D disponibles que sí ofrecen alta precisión y exactitud garantizada y a su vez son relativamente fáciles de usar se ofrecen a precios superiores a los U$D 20,000. Todo esto limita significativamente el acceso a esta tecnología a numerosos grupos de investigación con presupuestos limitados o falta de conocimiento técnico en computación. En esta tesis desarrolló e implementó un sistema de escaneo 3D de alta precisión y exactitud. El escáner se basa en visión estéreo utilizando luz estructurada. El mismo es integrado y de bajo costo con una interfaz de usuario que permite ser calibrado y utilizado por usuarios ajenos al campo. A su vez se desarrollaron dos técnicas de súper-resolución para lograr superar los límites de resolución impuestos por los dispositivos de bajo costo que componen el escáner. La primera está integrada al escáner y está basada en hardware, utilizando una plataforma de movimiento preciso para controlar la posición del objeto escaneado. Mientras que la segunda técnica consiste en un algoritmo de escaneo volumétrico por tallado (*space carving*). Tanto la implementación del escáner y como las técnicas de súper-resolución se evaluaron en su precisión, exactitud y calidad al escanear una serie objetos de prueba con garantía de fabricación. Se concluye que el escáner 3D desarrollado alcanza niveles de precisión y exactitud comparables con escáneres 3D que cuestan un orden de magnitud de precio más, aunque con un tiempo de adquisición mayor. A su vez el escáner logra alcanzar súper-resolución utilizando la técnica basada en hardware, mientras que la técnica de *space carving* necesita trabajo adicional para lograr generar superficies limpias a partir del tallado logrado.

Palabras claves: Escáner 3D, bajo costo, integrado, alta precisión y exactitud de captura, luz estructurada, súper-resolución, escaneo volumétrico, tallado (*space carving*).

A LOW COST EMBEDDED 3D SCANNING SYSTEM USING STRUCTURED-LIGHT AND SUPER-RESOLUTION

3D scanning has found widespread use in several engineering and scientific research areas. As of today, low-cost 3D scanners are either designed for research purposes and require advanced technical knowledge for their setup and use, or they are designed for use in games or other low precision applications and do not have a user interface, therefore requiring even more technical knowledge. On the other hand, highly accurate and precise 3D scanners that offer a seamless user experience are available at approximately U$D20.000 or more. All of this makes this technology inaccessible for research teams all around the globe that work on tight budgets or lack computer vision background knowledge. This thesis includes the development of a low-cost, embedded and easy to use 3D scanner that achieves high accuracy and precision. This scanner uses a low-cost stereo setup and solves the correspondence problem using structured light projected with a low-cost pico-projector. To go past the limits of resolution imposed by the low-cost projection device this work includes the development of two super-resolution techniques. The first one is integrated into the scanner and works by linearly translating the scanned object in controlled super-resolution steps during the capture. The second is complementary and purely algorithmic and generates a volumetric scanning of the object through space carving using the triangulated camera rays. Both the 3D scanner and super-resolution techniques developed in this thesis are tested and validated for the accuracy, precision and quality of the 3D models obtained. This is achieved by scanning reference objects with low error tolerance and statistically and visually analyzing the results. Based on the results obtained we conclude that the 3D scanner developed in this thesis achieves a level of accuracy and precision similar or in some cases better than professional 3D scanners available in the market at a fraction of the cost, but with a higher execution time. At the same time the scanner captures point clouds with super-resolution by using the hardware based super-resolution technique integrated into it, whereas the space carving technique needs additional work for the surface reconstruction phase.

Keywords: 3D Scanner, low-cost, embedded, high accuracy and precision, structured light, super-resolution, volumetric 3D scanning, space carving.

AGRADECIMIENTOS

A la Universidad de Buenos Aires y la Facultad de Ciencias Exactas y Naturales, especialmente al Departamento de Computación por permitirme acceder a una educación universitaria de excelente calidad.

A la comunidad del Departamento de Computación, que está llena de gente que, en muchos casos simplemente por amor al arte, nos ofrecieron su conocimiento y su ayuda a lo largo de la carrera.

A Pachi por estar siempre, su predisposición para hacer esto posible, y por bancarme por tanto tiempo. Y, aparte de lo profesional, una persona de oro.

A mis amigos, los que la vida y la carrera académica y profesional nos hizo cruzar caminos y crear amistades increíbles e invaluables. No me imagino una vida sin ustedes.

A mi familia que me bancaron y acompañaron en todo, los amo.

Al equipo de Visión, con el que compartimos tanto tiempo y la pase tan bien.

A Gabriel Taubin y su equipo de Brown del cual aprendí y recibí muchísima ayuda.

A Pablo Milla, Martín Hoqui, Sinapsis Tech S.A.S. y Moltech S.A. por dejarnos acceder a herramientas indispensables para la validación de esta tesis.

Índice general

1. INTRODUCCIÓN

1.1. Motivación

La demanda de sistemas de escaneo en tres dimensiones (3D) de alta precisión se ha visto incrementada en distintas áreas de investigación interdisciplinarias. Este fenómeno se vio potenciado debido a la popularización en los últimos años de varios sensores de captura 3D.

El primer sensor 3D de bajo costo, producido en masa y ofrecido al público general fue el sensor Kinect de Microsoft [SJP13], utilizado por la consola de videojuegos Xbox para capturar la posición de los usuarios. Más recientemente Apple empezó a incluir un sensor 3D llamado TrueDepth [BSK+19] en sus teléfonos (a partir del iPhone X específicamente) para autenticar al usuario a partir de un modelo 3D de su rostro.

Para ambos sensores se pusieron a disposición herramientas de desarrollo para explotar la información ofrecida por los mismos, y al abrirse esta tecnología al público se incrementó sustancialmente su aplicación en numerosas áreas de investigación, generando así una demanda de sistemas de escaneo 3D cada vez más precisos y accesibles.

Por un lado, la disponibilidad de esta clase de sensores logró llevar esta tecnología al alcance de muchos más grupos de investigación y hobbistas. Por otro lado, al ser sensores creados para un fin específico estos sensores no sólo son poco flexibles sino que tampoco tienen garantías de precisión y exactitud que son necesarias en entornos de investigación. Por ejemplo en el caso del sensor Kinect para distinguir siluetas a una distancia promedio de uso de un televisor y en el caso del sensor TrueDepth para capturar en distancias cortas los rostros de los usuarios.

Sensores 3D de alta precisión y exactitud para entornos más exigentes o productivos, existen desde hace más de 20 años pero sus costos son elevados, limitando el acceso a la tecnología.

1.2. Aplicaciones

Los sistemas de escaneo 3D actualmente son utilizados activamente por una inmensa gama de disciplinas de investigación y desarrollo.

En el marco industrial y de ingeniería mecánica se utiliza por ejemplo para el control de calidad en la fabricaciones de partes en plástico y metal [VGB]; en el análisis de estrés y deformación de materiales [BAT13]; al medir objetos frágiles o calientes; en la digitalización de partes discontinuas o antiguas para ingeniería inversa; para digitalizar prototipos [JWK+13]; entre otros.

En medicina se utiliza ampliamente en odontología para el análisis y posterior corrección de dentaduras; para medir la forma de la espalda de pacientes con escoliosis y lograr un seguimiento; al capturar y estudiar la topografía de la piel; para capturar una extremidad al fabricar prótesis logrando encajes casi perfectos al cuerpo del paciente; entre otros [ARV+14] [PCL+13].

En el campo arqueológico se utiliza ampliamente para la captura y catalogado de restos hallados [KS08], permitiendo la manipulación y rearmado de estructuras de forma digital sin la necesidad de arriesgar la integridad de los objetos encontrados.

Por último, con el reciente auge de las impresoras 3D, el escaneo 3D ha comenzado a utilizarse para replicar piezas y diseñar iterativamente.

Los sistemas de escaneo utilizados en un ámbito industrial y en la investigación, no sólo deben ofrecer alta precisión y exactitud, sino que es aún más importante proveer de garantías sobre el error de las mediciones. Adquirir sistemas que cumplen estos requerimientos actualmente conlleva una alta inversión de capital.

Una amplia gama de escáneres 3D se pueden adquirir hoy en día, los precios dependen generalmente de la calidad de los resultados obtenidos y el control que se pueda tener del dispositivo.

Los sensores Kinect tienen un costo de entre U$D100 y U$D200, y si bien no fueron creados con fines de investigación, se han realizado exhaustivos estudios para medir su precisión, evidenciando que son muy vulnerables al entorno, ya sea luz o temperatura, logrando baja y poco consistente precisión [WS16]). Para el sensor TrueDepth de Apple, el costo es el del teléfono entero, ya que no está disponible por su cuenta, es decir U$D1000 mínimo, y su precisión no ha sido analizada académicamente aún.

Otro escáner para uso de hobbistas pero específicamente diseñado para desarrolladores es el sensor de Occipital que puede ser conectado a un iPad o iPhone a un costo de U$D400 dólares, con una exactitud reportada en los 3 mm cuando el objeto se encuentra cerca del sensor, pero agrega el hecho de usar un dispositivo intermediario (un iPhone o iPad), limitando así su usabilidad.

Apuntando a sensores profesionales y subiendo un orden de magnitud en la precisión, se puede adquirir un sensor fabricado por Rodin4D para fines médicos a unos U$D10.000 que logra una exactitud reportada de 0,5 mm.

Finalmente subiendo aún más el orden de precisión tenemos los sensores desarrollados por Minolta con una exactitud reportada en 0,05 mm y un costo de al menos U$D20000, y HandyScan 3D 300 desarrollado por Creaform con un una exactitud de 0,025 mm y costo estimado de más de U$D40.000.

1.3. Métodos de escaneo 3D

El escaneo 3D consiste en capturar la forma de algún objeto del mundo real para luego generar un modelo 3D digital que lo represente. El proceso de escaneo se puede ver como un *pipeline* que consiste de los siguientes pasos.

Primero, la adquisición de datos del objeto a analizar. Como segundo paso, la generación de una nube de puntos a partir de esos datos. Como tercer paso la estimación de la superficie del objeto escaneado; y por cuarto y último paso la representación de esta superficie como una malla poligonal.

En la actualidad existen numerosos métodos utilizados para capturar la forma de un objeto. Estos métodos se pueden categorizar dependiendo la interacción que tiene con el objeto a analizar. Principalmente se van a dividir en dos: los métodos de contacto y los métodos sin contacto con el objeto.

Los métodos de contacto se basan en obtener puntos de la superficie del objeto al tocarlo con un instrumento de medición. Por ejemplo, utilizando un brazo articulado con punta sensorial. El mismo, puede ser operado tanto manual como automáticamente y al detectar contacto con una superficie se utilizan sensores posicionales en las articulaciones para grabar la coordenada de la punta al momento del contacto. Uno de los dispositivos que realiza esta medición de forma autónoma tiene el nombre de *Coordinate Measuring*

Machine[1].

Dentro de los métodos sin contacto podemos encontrar dos categorías principales: pasivos y activos. En general, los métodos pasivos sin contacto se basan en capturar imágenes del objeto a analizar, para luego inferir su forma mediante las mismas, mientras que los métodos activos sin contacto se basan en emitir una señal y capturar el reflejo de esta señal en el objeto para estimar su forma. Un buen resumen de los métodos pasivos y activos fue realizado por Giancola et al. en [GVS18]. Los métodos de escaneo 3D por luz estructurada entran en la categoría de métodos activos sin contacto. El método utilizado en esta tesis utiliza luz estructurada de patrones temporales Gray emitidos por un proyector digital y será descripto de forma detallada más adelante.

A partir de los datos capturados por cualquiera de estos métodos es necesario en muchos casos generar una superficie. Una superficie permita entre otras cosas dar un límite claro entre el interior y el exterior del objeto escaneado. Para ello existen múltiples algoritmos de reconstrucción de superficie, entre los que se destacan *Poisson surface reconstruction* [KBH06] junto con su variante *Screened Poisson surface reconstruction* [KH13].

Un punto a observar es que todos estos métodos de captura (tanto pasivos como activos) están limitados en resolución por los sensores a utilizar, en nuestro caso las cámaras y el proyector. Es por esto que existen técnicas de súper-resolución para vencer estas limitaciones. La súper-resolución consiste en alcanzar resoluciones más altas que las provistas por los sensores o métodos de iluminación disponibles cuando corresponda. En los métodos de captura activos, las técnicas de súper-resolución se pueden aplicar para incrementar la resolución del sensor de captura, o para incrementar la resolución del dispositivo de iluminación.

1.4. Objetivos

El objetivo principal de esta tesis es estudiar, diseñar e implementar un sistema de escaneo 3D integrado y de bajo costo.

Se pretende lograr que el sistema alcance exactitud y precisión comparable a los sistemas de escaneo 3D disponibles en el mercado. Para esto se trabajará en el método de escaneo por luz estructurada, sobre el cual se desarrollarán y aplicarán técnicas de súper-resolución basadas tanto en hardware como en software.

Un objetivo importante que se busca lograr es un diseño modular del sistema que facilite la expansión y alteración de los componentes para futuros desarrollos. Como objetivo adicional se desarrollará una técnica basada en tallado espacial (*space-carving*) utilizando la información geométrica devuelta por la técnica de luz estructurada, con el objetivo similar de lograr súper-resolución en el modelo generado.

Por último, al buscar lograr un sistema integrado se desarrollará una interfaz de usuario que permita preparar y utilizar el sistema remotamente y de forma sencilla.

1.5. Trabajo relacionado

A lo largo de los últimos años, se han desarrollado numerosos sistemas de escaneo 3D. Un análisis de varios escáneres comerciales puede verse en Redaelli et al. [RBF+18] y también en Straub et al. [SKMK15].

[1] `https://en.wikipedia.org/wiki/Coordinate-measuring_machine`

Soluciones de bajo costo también han sido desarrolladas. Las mismas, suelen estar basadas en métodos de captura sin contacto con iluminación pasiva. También, existen otros con mayor exactitud que utilizan iluminación activa mediante la proyección de planos láser y bandejas giratorias de precisión. Entre las soluciones de bajo costo, se pueden destacar: el escáner llamado FabScan desarrollado por Mario Lukas [Luk15], que consiste en una Raspberry Pi, una cámara web, una bandeja giratoria y un láser LED que proyecta una línea. Todo esto dentro de un compartimiento para bloquear la luz externa. En su trabajo, también se puede leer un análisis muy completo de soluciones de bajo costo anteriores.

Un análisis de las desventajas en velocidad y calidad que sufren los escáneres de este tipo, fue llevado a cabo por Kedzierskia et al. [KWFC16], centrándose en el FabScan. Se concluyó que la exactitud alcanzada por estas soluciones va desde los 2.9mm a los 9mm para superficies con muchos detalles.

Otro escáner pensado como una solución de bajo costo, *open-source* y *do-it-yourself* (DIY) es el denominado *Free LSS Scanner* [LSS19] el cual tiene una comunidad de desarrollo muy activa. Este escáner es similar al *FabScan*, y consiste en utilizar un láser adicional.

Ambas soluciones son integradas y capturan de forma automática, con una Raspberry Pi sirviendo una aplicación web que proporciona una interfaz intuitiva y fácil de usar para operar el escáner y visualizar los resultados.

Con respecto a los métodos de súper-resolución se han desarrollado múltiples técnicas a lo largo del tiempo. Para obtener súper-resolución con métodos de captura activos de luz estructurada Weinmann et al. [WSRK11] desarrolló un método en el cual se proyecta la luz estructurada desde varias ubicaciones, combinando los resultados y por lo tanto refinando las correspondencias. Otra técnica fue desarrollada por investigadores de Microsoft en [Cor13] y consiste en vibrar la fuente de luz estructurada de forma controlada para luego encontrar el centro de cada correspondencia de forma precisa.

1.6. Estructura de la tesis

La tesis está compuesta por cuatro capítulos principales: “Preliminares”, “Desarrollo”, “Resultados” y “Conclusiones y Trabajo Futuro”.

En “Preliminares” se presentarán todas las técnicas y modelos matemáticos utilizados en el desarrollo y evaluación del escáner desarrollado y los métodos de súper-resolución. El capítulo de “Desarrollo” consistirá de una presentación detallada del sistema de escaneo 3D y los métodos de súper-resolución desarrollados. En el capítulo de “Resultados” se buscará evaluar formalmente la calidad de todo lo desarrollado en esta tesis. A su vez se va a comparar el sistema 3D desarrollado mano a mano con un escáner profesional *HandyScan 3D 300* desarrollado por $Creaform$[2]. Por último en “Conclusiones y Trabajo Futuro” se hará un resumen de lo logrado en esta tesis, y de las oportunidades de mejora en las que se pueda trabajar a futuro.

[2] `https://www.aniwaa.com/product/3d-scanners/creaform-handyscan-300/`

2. PRELIMINARES

En este capítulo vamos a introducir los conceptos necesarios para entender cómo funciona la captura 3D. Vamos a concentrarnos en los conceptos y modelos utilizados específicamente en los escáneres 3D por luz estructurada.

En un principio definamos qué es el escaneo por luz estructurada. El escaneo por luz estructurada es un método de captura 3D basado en triangulación que utiliza luz estructurada para resolver el problema de correspondencia.

Que utiliza triangulación significa que va a calcular la distancia al objeto escaneado mediante trigonometría. Esto significa que va a observarlo desde múltiples ubicaciones conocidas, y luego utilizar los ángulos de visión para formar un triángulo, calculando así la ubicación del objeto.

Para lograr esto en necesario reconocer el objeto al observarlo desde las distintas ubicaciones, para poder calcular dichos ángulos de visión. Éste es el llamado problema de correspondencia, ya que decide qué puntos de las múltiples observaciones se corresponden entre sí.

Es en el problema de correspondencia que entra en juego la luz estructurada. La luz estructurada es uno o varios patrones de luz que son proyectados sobre la escena, y al capturar una imagen de dicha escena estos pueden ser reconocidos y ubicados de forma algorítmica. Al ser reconocidos y ubicados en varias observaciones, uno puede generar la correspondencia entre dichas observaciones.

Por supuesto en nuestro escáner, dado que trabajamos en visión computacional, una observación va a ser una imagen bidimensional capturada con una cámara digital. Por otro lado la luz estructurada es proyectada utilizando un proyector digital. Tanto la imagen capturada y como la imagen proyectada, al ser digitales consisten en una matriz de píxeles de dos dimensiones.

Dado este contexto a continuación vamos a presentar los conceptos nombrados y los modelos en los que se basan, de forma detallada. A su vez desarrollaremos el concepto de luz estructurada, profundizando particularmente en los métodos utilizados en esta tesis.

Una vez dados los modelos explicaremos cómo ajustamos un sistema en particular a dichos modelos mediante métodos de calibración.

Luego vamos a mostrar cómo se captura un objeto 3D utilizando el sistema y cómo se procesa dicha captura para generar superficies 3D.

Por último, al ser un escáner un instrumento de medición, presentaremos cómo se caracteriza un instrumento de medición para luego poder evaluar el escáner desarrollado y compararlo con otros escáneres disponibles.

2.1. Imágenes y Triangulación

En esta sección explicaremos cómo se forma una imagen utilizando tanto una cámara o proyector en general, y en el mundo digital en particular. Presentaremos los modelos matemáticos que describen el proceso de formación de imagen, como se ajustan dichos modelos una cámara en particular. Presentaremos a su vez como se modela un escáner 3D de triangulación y el proceso de triangulación en sí.

2.1.1. Formación de imagen

Una cámara trabaja capturando la luz reflejada por la escena que está observando. Esta luz atraviesa su lente y alcanza una película analógica, o un sensor de luz electrónico. Esto es esencialmente una proyección del mundo 3D en un *plano de imagen* 2D, ya sea el sensor o la película y tiene el nombre de *proyección perspectiva*.

Un modelo matemático simple que se utiliza para describir este proceso es el **modelo de cámara estenopeica ideal** (Figura 2.1).

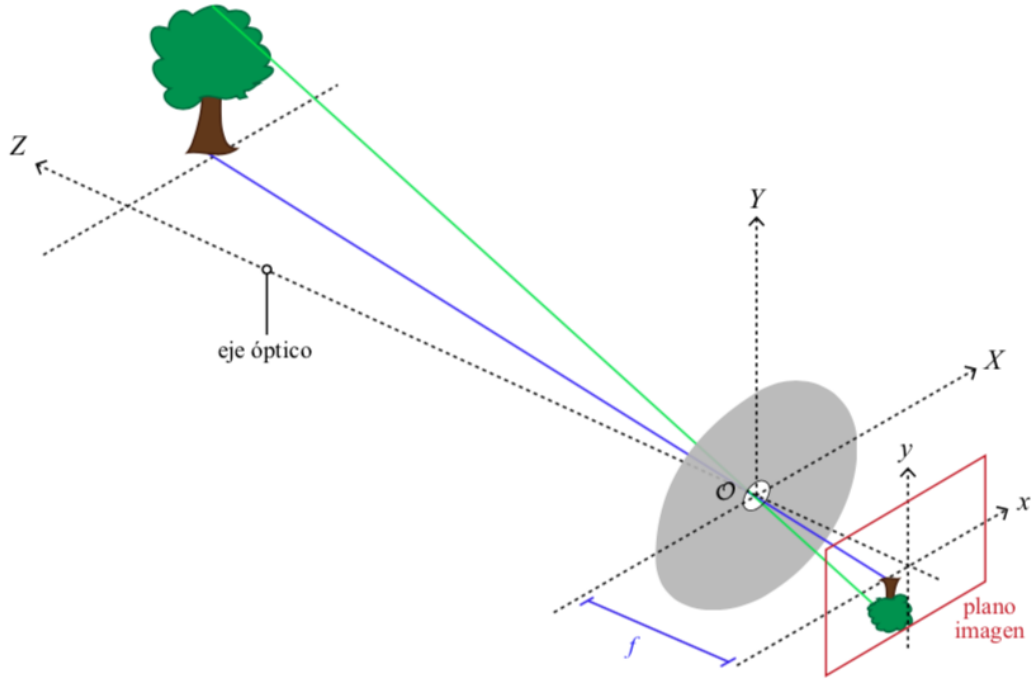


Fig. 2.1: Principios del modelo de cámara estenopeica ideal.

El modelo de cámara estenopeica ideal describe la relación entre un punto en el espacio 3D, y su proyección en el plano de imagen. Este modelo es simple ya que describe una cámara estenopeica ideal. Es decir que se considera a la apertura de la cámara como un punto, y no se considera el uso de lentes.

El modelo consiste en un sistema de coordenadas 3D con origen en el centro óptico o *punto de apertura* de la cámara O y un plano paralelo a dos ejes del sistema de coordenadas llamado *plano imagen*. La distancia entre el plano y el punto de apertura f se llama *distancia focal*. Todo punto en el espacio va a generar una única línea que también contiene al punto de apertura y, al menos que la línea sea paralela al plano de imagen, esta interseca al plano de imagen en un único punto.

Si reemplazamos esta línea por un rayo, se puede modelar tanto una cámara como un proyector. Por convención en una cámara los rayos se originan en la escena 3D observada, con dirección al punto de apertura de la cámara. Mientras que en un proyector, el rayo se origina en el plano imagen con dirección hacia el punto de apertura del proyector.

Notemos en la Figura 2.1 que al pasar por el punto de apertura, los rayos se cruzan tanto horizontal como verticalmente. Esto resulta en una imagen rotada 180 grados. Para

representar de forma más natural e intuitiva lo que sucede en la vida real podemos modificar levemente el modelo. Para esto ubicamos el plano imagen a distancia f del punto de apertura en vez del $-f$ original, obteniendo la misma proyección pero sin la rotación de 180 grados (Figura 2.2), a este plano se lo conoce como *plano imagen virtual*. A partir de ahora siempre que hablemos de plano imagen estaremos hablando del plano imagen virtual.

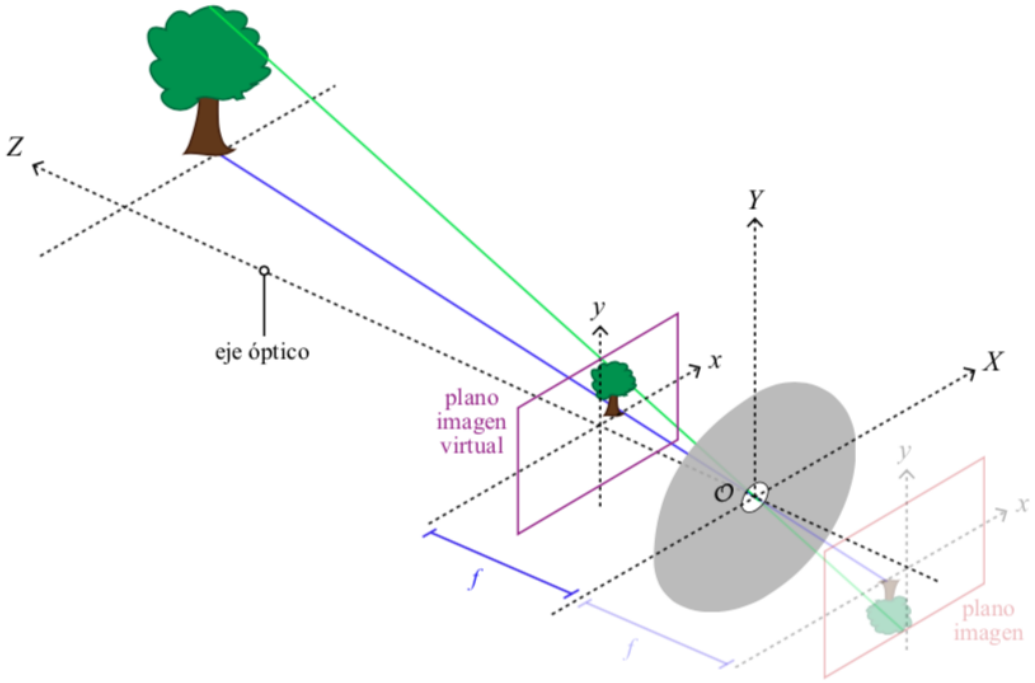


Fig. 2.2: Modelo de cámara estenopeica utilizando plano virtual.

En el modelo modificado el origen del sistema va a llamarse **centro de proyección**. Formalmente el modelo (Figura 2.3) consiste de:

- Un sistema de coordenadas 3D ortogonal con origen en el centro de proyección O llamado *sistema de coordenadas de cámara*. El eje Z apunta en dirección de captura de la cámara, por lo tanto se lo llama *eje óptico, eje principal* o *rayo principal*.
- El plano imagen, donde se proyecta la escena 3D. Este plano es paralelo a los ejes X e Y del sistema de coordenadas de cámara, a una distancia f del centro de proyección, siendo f la distancia focal mencionada previamente.

A su vez podemos observar los siguientes elementos:

- El punto R donde interseca el eje óptico y el plano imagen, llamado *centro de imagen* o *punto principal*.
- Un sistema de coordenadas 2D en el plano imagen con origen en R llamado *sistema de coordenadas de imagen*.

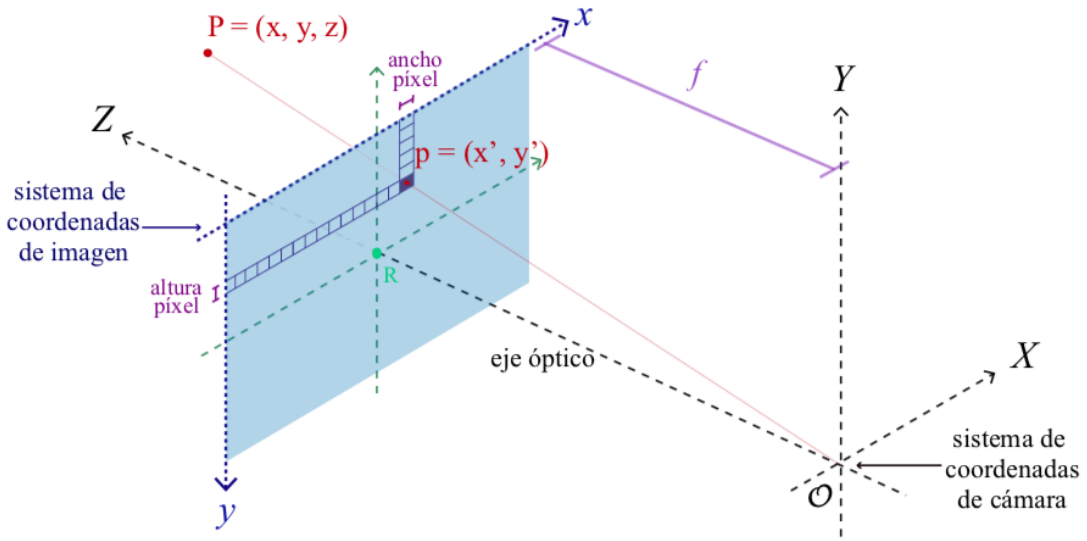


Fig. 2.3: Modelo de cámara estenopeica ideal formal.

Si consideramos un punto 3D P con coordenadas $(x, y, z)^t$ vemos que la línea de proyección pasa por P y por O. Esta interseca el plano imagen en el punto $p = (x', y')^t$, siempre y cuando la línea no sea paralela al plano.

El modelo nos permite calcular p a partir de P como se puede ver en la Ecuación 2.1. Esta relación de $\mathbb{R}^3$ en $\mathbb{R}^2$ es llamada transformación proyectiva. Esta relación es válida mientras se asuma que la línea de proyección no es paralela al plano imagen, lo cual es válido en la práctica ya que la escena observada se encuentra delante de la cámara, $z > 0$.

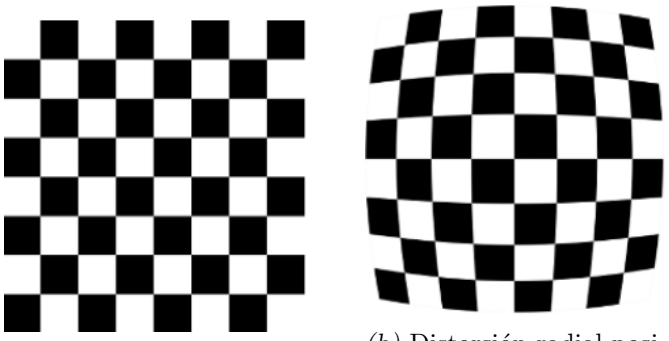
$$p = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} f\frac{x}{z} \\ f\frac{y}{z} \end{pmatrix} \tag{2.1}$$

Al estar modelando una cámara estenopeica ideal, este modelo no refleja correctamente lo que sucede en una cámara real, menos aún en una digital. Por un lado las cámaras utilizan lentes para enfocar los rayos de luz. Estas lentes no solo son imperfectas, sino que no están perfectamente alineadas a la película o el sensor de la cámara. La imperfección de la lente introduce distorsión radial, mientras que la mala alineación introduce distorsión tangencial (Figura 2.4).

Además de esto, en las cámaras digitales encontramos más variaciones. Primero, el origen del sistema de coordenadas de la imagen varía, generalmente ubicándose en la esquina superior izquierda de la imagen. Segundo, las imágenes digitales están divididas en píxeles discretos no necesariamente cuadrados. Por último, el sensor digital puede introducir no-linealidad distorsionando la relación.

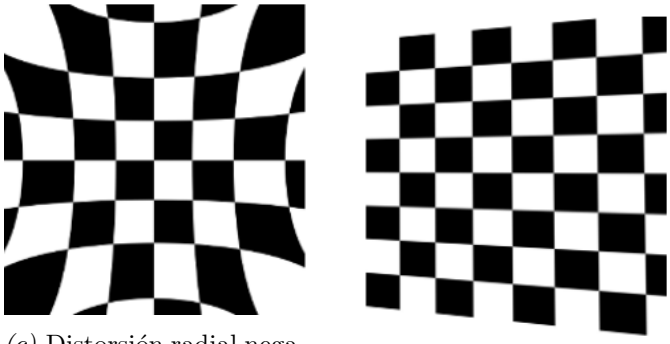
Para considerar todas estas diferencias se introducen una serie de transformaciones adicionales al modelo.

Primero agregamos una translación $(c_x, c_y)^t$ para corregir el cambio de origen del sis-



(a) Imagen original.

(b) Distorsión radial positiva.



(c) Distorsión radial negativa.

(d) Distorsión tangencial.

Fig. 2.4: Tipos de distorsión óptica.

tema de coordenadas de imagen:

$$p = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} f\frac{x}{z} + c_x \\ f\frac{y}{z} + c_y \end{pmatrix} \tag{2.2}$$

Para considerar la discretización en píxeles se introducen dos parámetros, k y l. Estos corresponden al alto y ancho de cada píxel respectivamente (ya que el píxel puede no ser cuadrado) para la unidad en la que queramos modelar el espacio:

$$p = \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} fk\frac{x}{z} + c_x \\ fl\frac{y}{z} + c_y \end{pmatrix} \tag{2.3}$$

Por ejemplo en caso de utilizar el sistema métrico, la unidad de k y l podría ser $pixeles/milimetro$.

Para continuar con las demostraciones va a ser útil representar esta transformación como un producto de una matriz por el vector de entrada (el punto 3D). El problema es que como vemos en la Ecuación 2.3 la proyección no es lineal, ya que el la operación divide uno de los parámetros de entrada, en nuestro caso z.

Una forma de evitar este problema es llevar la transformación al sistema de coordenadas homogéneo. Para esto simplemente agregamos una dimensión tanto al vector de entrada y como al de salida, con un 1 en la nueva dimensión. Notemos que la igualdad entre un vector y sus coordenadas homogéneas solo ocurre cuando la última coordenada es 1. Por lo

tanto, al convertir una coordenada homogénea $(v_1, ..., v_n, w)^t$ obtenemos las coordenadas Euclidianas $(\frac{v_1}{w}, ..., \frac{v_n}{w})^t$. Usando las coordenadas homogéneas podemos formular:

$$p_h = \begin{pmatrix} fkx + c_x z \\ fly + c_y z \\ z \end{pmatrix} = \begin{pmatrix} fk & 0 & c_x & 0 \\ 0 & fl & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} fk & 0 & c_x & 0 \\ 0 & fl & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} P_h \tag{2.4}$$

Desde este punto vamos a asumir que trabajamos con coordenadas homogéneas. Como vemos en la Ecuación 2.4 podemos representar la relación entre el punto 3D P y sus coordenadas p en la imagen como un producto matriz-vector:

$$p = \begin{pmatrix} x' \\ y' \\ z \end{pmatrix} = \begin{pmatrix} fk & 0 & c_x & 0 \\ 0 & fl & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} fk & 0 & c_x & 0 \\ 0 & fl & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} P = MP \tag{2.5}$$

Si la descomponemos aún más:

$$p = MP = \begin{pmatrix} fk & 0 & c_x \\ 0 & fl & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} I & 0 \end{pmatrix} P = K \begin{pmatrix} I & 0 \end{pmatrix} P \tag{2.6}$$

A la matriz K la llamamos **matriz de cámara** (*camera matrix*) ya que describen a la cámara digital. La coordenada en el plano imagen $u = (u_1, u_2)$ se obtiene de la siguiente manera:

$$u = \begin{pmatrix} \frac{x'}{z} \\ \frac{y'}{z} \end{pmatrix} \tag{2.7}$$

Aún nos faltan dos parámetros para terminar de describir una cámara digital real de forma fiel: el **desvío** (*skewness*) y la **distorsión**.

Decimos que una imagen está desviada cuando el sistema de coordenadas de la cámara está desviado. Es decir que el ángulo entre los ejes es levemente mayor o menor a 90 grados. Si bien la mayoría de las cámaras no tienen desvío, hay casos que por errores de fabricación el sensor de la cámara puede introducir cierto desvío. Para contemplarlo agregamos un ángulo de desvío θ a K de la siguiente manera:

$$\alpha = fk \tag{2.8}$$

$$\beta = fl \tag{2.9}$$

$$K = \begin{pmatrix} \alpha & -\alpha \cot \theta & c_x \\ 0 & \frac{\beta}{\sin \theta} & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.10}$$

Por último, la distorsión de lente se puede modelar de muchas maneras, la más utilizada fue introducida en [HS97] y consiste en utilizar 3 parámetros k_1, k_2 y k_3 para la distorsión radial y 2 adicionales p_1 y p_2 para la distorsión tangencial, como se puede ser a continuación:

$$P' = \begin{pmatrix} X' \\ Y' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \\ 1 \end{pmatrix} \tag{2.11}$$

$$P_d = \begin{pmatrix} X_d \\ Y_d \\ 1 \end{pmatrix} = \begin{pmatrix} X'(1 + k_1 r^2 + k_2 r^2 + k_3 r^6) + 2p_1 X'Y' + p_2(r^2 + 2X'^2) \\ Y'(1 + k_1 r^2 + k_2 r^2 + k_3 r^6) + p_1(r^2 + 2Y'^2) + 2p_2 X'Y' \\ 1 \end{pmatrix} \tag{2.12}$$

$$r^2 = X'^2 + Y'^2 \tag{2.13}$$

$$p = MP_d \tag{2.14}$$

Hasta aquí tenemos un modelo de cámara que nos permite modelar una cámara digital del mundo real, y relacionar puntos del mundo 3D con la imagen 2D (plano imagen) capturada por dicha cámara. Todo esto en un sistema de coordenadas que se origina en el centro de proyección de dicha cámara. A este modelo lo llamamos modelo de cámara estenopeica general.

¿Pero qué sucede si la información del mundo 3D está disponible en otro sistema de coordenadas? Vamos a necesitar contemplar una transformación adicional que relacione coordenadas del sistema de coordenadas del mundo, con el sistema de coordenadas de la cámara. Esta transformación está formada por una matriz de rotación R y un vector de translación T. Por lo tanto, dado un punto 3D del sistema de coordenadas del mundo P_w podemos calcular sus coordenadas en el sistema de la cámara de la siguiente manera:

$$P = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} P_w \tag{2.15}$$

A este punto ya tenemos un modelo completo que nos permite ir de un punto 3D de un sistema de coordenadas arbitrario al plano imagen de una cámara. Se puede ver que la matriz de proyección M contiene dos tipos de parámetros: **Intrínsecos** y **Extrínsecos**. Todos los parámetros que están dentro de la matriz K junto con los parámetros de distorsión son los intrínsecos, ya que son específicos a una cámara en particular. Mientras que los extrínsecos son la rotación y translación, ya que posiciona la cámara en un sistema de coordenadas externo.

2.1.2. Calibración

Calibrar consiste en ajustar los modelos matemáticos mencionados a una configuración de cámaras real. Es decir, estimar los parámetros intrínsecos y extrínsecos del sistema que vimos antes. Lograremos esto resolviendo la matriz K junto con parámetros de distorsión para obtener la calibración intrínseca y la matriz de rotación R junto con el vector de translación T para la calibración extrínseca.

Si bien a lo largo del tiempo se han desarrollado muchos métodos de calibración, una de las técnicas más utilizadas es la presentada por Zhang et al. en [Zha99]. La técnica consiste en capturar un tablero plano similar al de un juego de ajedrez (Figura 2.5) en múltiples posiciones (Figura 2.6).

Hay algoritmos que permiten encontrar las esquinas internas de un tablero de estas características de forma precisa y robusta. Conociendo las dimensiones internas del tablero se generan coordenadas 3D reales de las esquinas internas en un sistema de coordenadas

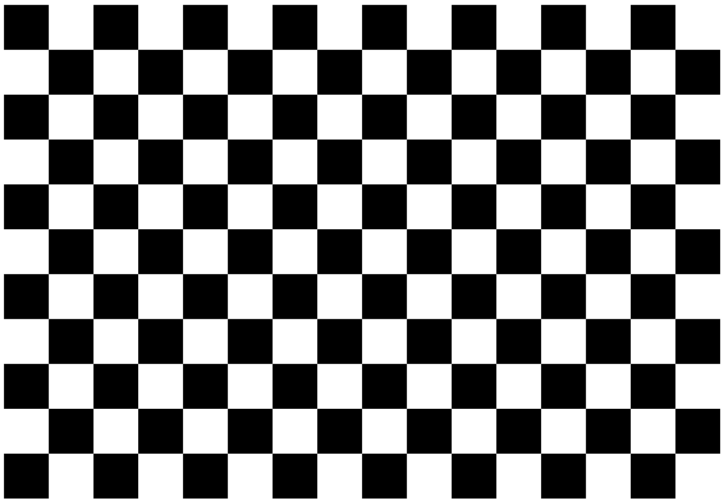


Fig. 2.5: Tablero de calibración.

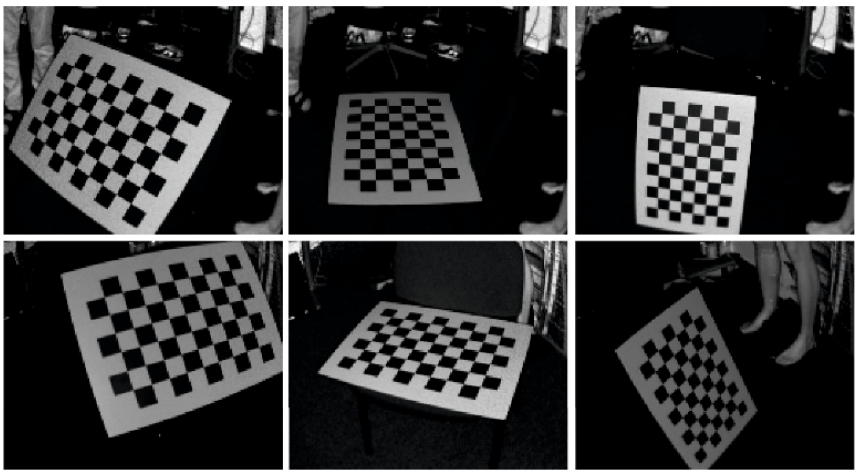


Fig. 2.6: Capturas del tablero en diferentes posiciones.

con origen en el tablero como se ve en Figura 2.7. La unidad que se utilice en este sistema van a ser finalmente las unidades en las que trabajará el modelo.

Se procede detectando estas esquinas internas en las imágenes capturadas y generando pares de correspondencias entre los puntos donde se detectaron las esquinas, y las coordenadas 3D generadas previamente. La idea general consiste en utilizar estas correspondencias para crear un sistema de ecuaciones dado por Ecuación 2.14 y Ecuación 2.15. Se usa este sistema de ecuaciones para obtener los parámetros intrínsecos que mejor se ajusten a las correspondencias.

Como el plano del tablero y su imagen están relacionados por una homografía, se puede utilizar la homografía como restricción para crear un sistema de ecuaciones que nos permite calcular la matriz K (Ecuación 2.10) del modelo de cámara estenopeica sin distorsión. Para esto se necesitan al menos 3 capturas del plano.

Con esta matriz K inicial, se procede a incluir los parámetros de distorsión de lente y generar un nuevo sistema de ecuaciones, esta vez buscando calcular los parámetros de

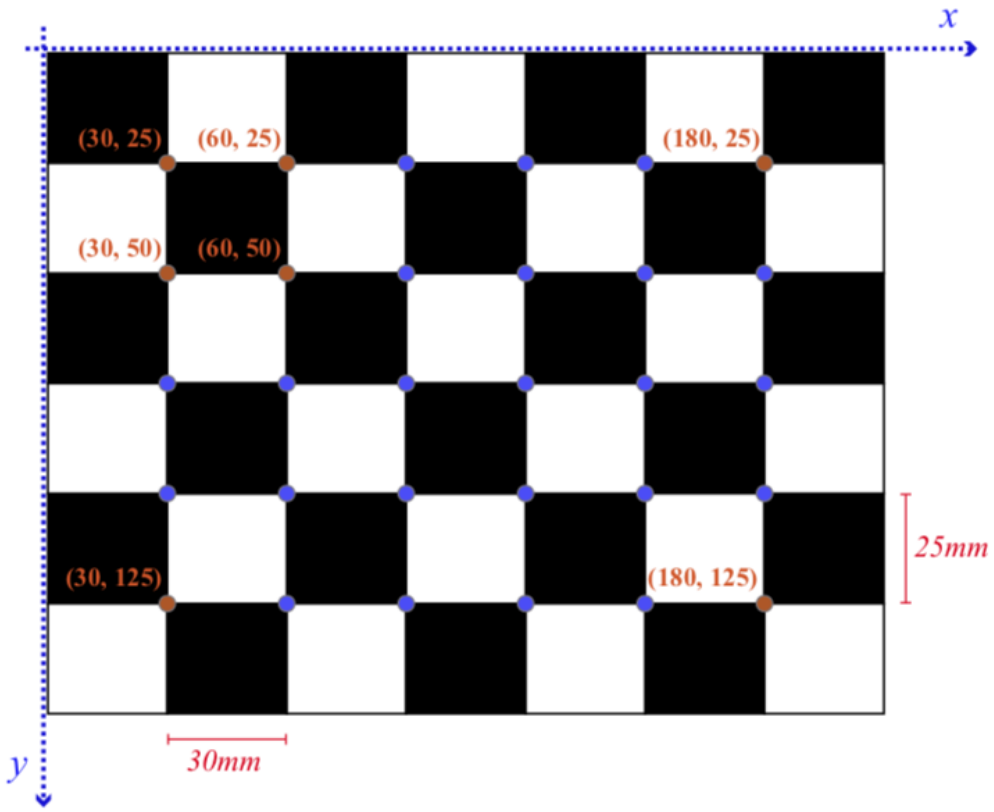


Fig. 2.7: Coordenadas de las esquinas internas del tablero. En rojo se puede ver el alto y ancho de la cuadricula, mientras que en naranja se genera puede ver como se genera la coordenada de ciertas esquinas de forma representativa.

distorsión y optimizar la matriz de cámara K. Para lograr esto se busca obtener la matriz K y parámetros de distorsión que minimicen el error de reproyección de las esquinas del tablero en el plano imagen del modelo.

El error de reproyección se calcula proyectando las esquinas del tablero en el plano imagen del modelo ajustado y midiendo su distancia con los puntos de imagen donde se detectaron efectivamente dichas esquinas. Para calcular dicho error primero se estima la pose (rotación y translación) del tablero con respecto a la cámara (utilizando la matriz K inicial), para cada imagen capturada (Figura 2.8). Dada esta pose se puede proyectar cada esquina del tablero en el plano imagen utilizando la Ecuación 2.7. Siendo p_{Ii} el punto de imagen donde se detectó la esquina i, y siendo p_{Ri} la proyección de la esquina i en el plano imagen, la distancia entre p_{Ii} y p_{Ri} es el error de reproyección.

Utilizando el método de máxima verosimilitud inicializado con la matriz K calculada antes, estimaremos los parámetros del modelo de cámara estenopeica general que minimice la suma de errores de reproyección (*reprojection error*[1]) del sistema.

Una vez calculados los mejores parámetros de calibración intrínseca para cada cámara/proyector del sistema se procede a calcular la translación y rotación de dichos dispositivos a un sistema de coordenadas de referencia. En la práctica suele utilizarse el sistema de coordenadas de alguno de los dispositivos como sistema de referencia.

[1] *Learning OpenCV 3*: Capitulo 18 [Bra16]

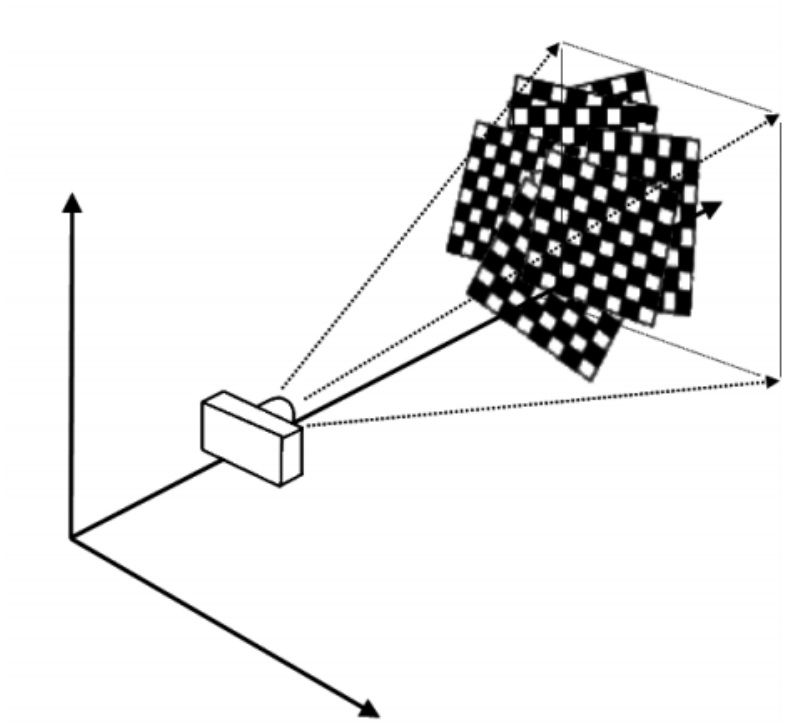


Fig. 2.8: Poses del tablero en las distintas imágenes.

Para lograr esto se captura el tablero en una única posición desde todos los dispositivos. Detectando el tablero se generan las correspondencias al igual que antes. Esta las utilizamos para calcular la pose del tablero respecto a cada cámara C_i, es decir su rotación R_{C_i} y translación t_{C_i} como hicimos antes. Sabiendo esto, de manera simple podemos calcular la rotación $R_{C_iC_0}$ y translación $t_{C_iC_0}$ entre cada dispositivo y el dispositivo de referencia C_0.

Sin pérdida de generalidad veamos como funciona para dos cámaras C_0 y C_1 designando a C_0 como dispositivo de referencia. Teniendo la pose del tablero respecto a la cámara C_0 y C_1: (R_0, T_0) y (R_1, T_1) respectivamente, podemos calcular $R_{C_1C_0}$ y $T_{C_1C_0}$ tal que:

$$R_1 = R_{C_1C_0} * R_0T_2 = R_{C_1C_0} * T_0 + T_{C_1C_0} \tag{2.16}$$

$R_{C_1C_0}$ junto a $T_{C_1C_0}$ nos va a permitir llevar cualquier rayo o punto en el sistema de coordenadas de C_1 al sistema de coordenadas de C_0.

En resumen, el proceso de calibración consiste en capturar un tablero de dimensiones conocidas al menos 3 veces desde cada cámara del sistema para calcular sus parámetros intrínsecos. Luego se captura al menos una vez el tablero en una posición fija desde todas las cámaras para calcular la posición relativa entre cámaras. Y para medir cuán buena es la calibración se utiliza el error de reproyección.

Aunque este es el mínimo de capturas a realizar, cuantas más capturas del tablero en distintas poses se realicen, más fiel será el modelo ajustado gracias al proceso de optimización.

El mismo proceso de calibración se puede realizar para un proyector adaptando el proceso de detección de esquinas del proyector y el generado de correspondencias, ya que el

proyector no permite capturar una imagen en las que se puedan detectar dichas esquinas. Una técnica para lograrla fue desarrollada por Moreno et al. en [MT12]. La técnica se basa en colocar el tablero dentro del campo de proyección del proyector, proyectar luz estructurada sobre el mismo, y capturar el resultado con una cámara cualquiera. En resumen, se detecta al igual que antes las esquinas del tablero en las imágenes capturadas y al mismo tiempo la decodificación de la luz estructurada permite saber qué píxel del proyector cubre cada una de las esquinas, generando la correspondencia entre píxel del proyector y esquina del tablero. Dada esta correspondencia el proceso de calibración continua igual que para una cámara.

2.1.3. Reconstrucción por triangulación

El trabajo en esta tesis se basa en determinar la ubicación 3D de un punto en el espacio utilizando su proyección en varias imágenes. Esto se puede hacer ya que la luz viaja por el aire en línea recta (en situaciones normales) y, como vimos antes, cada punto en una imagen está formado por un único rayo 3D que podemos calcular gracias al modelo de cámara estenopeica general.

A grandes rasgos esto nos permite recuperar la ubicación de un punto 3D intersecando los rayos formados por la proyección del punto 3D en las distintas imágenes (Figura 2.9).

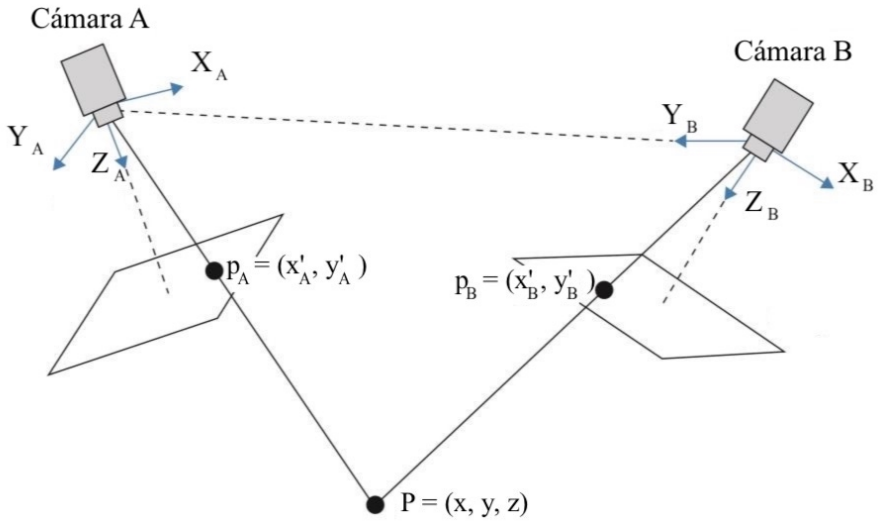


Fig. 2.9: Ejemplo de triangulación con dos cámaras.

En esta subsección desarrollaremos los conceptos necesarios para entender esta técnica.

Por lo visto anteriormente, al calibrar el modelo de cámara estenopeica general a una cámara digital, podemos calcular que rayo de luz compone a cada píxel de una imagen que la cámara capture. También vimos cómo modelar un conjunto de cámaras/proyectores en un único sistema de coordenadas. Dado un conjunto de cámaras calibradas tanto intrínseca como extrínsecamente en un mismo sistema de coordenadas, si un punto 3D P del espacio es observado por varias cámaras entonces va a proyectarse en el plano imagen de cada una. Para cada plano imagen podemos utilizar el píxel p_{Ii} donde es observado P y calcular el rayo que atraviesa dicho píxel. Luego, intersecando dichos rayos, se podría recuperar la coordenada 3D de P.

Primero veamos como se define una línea y un rayo.

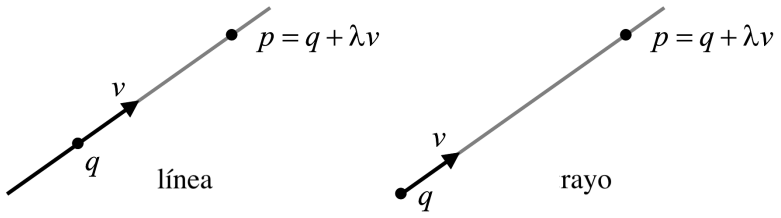


Fig. 2.10: Representación paramétrica de líneas y rayos.

Una línea se puede definir con uno de sus puntos q y su vector dirección v (Figura 2.10).

Cualquier otro punto p en la línea va a poder calcularse al sumar un escalar del vector dirección al punto q. El escalar λ puede ser positivo, negativo o cero. Esto da a lugar a la representación paramétrica de una línea:

$$L = \{\ p = q + \lambda v\ :\ \lambda \in \mathbb{R}\ \}\ . \tag{2.17}$$

Para la línea definida, esta representación no es única. Cambiando el punto q por cualquier otro punto en la línea, o cambiando el vector v por un escalar de norma distinta a 0 seguimos representando la misma línea.

Un rayo es la "mitad" de una línea, y se representa igual que una línea (Figura 2.10), con la diferencia que λ no puede tomar valores negativos:

$$R = \{\ p = q + \lambda v\ :\ \lambda \geq 0\ \} \tag{2.18}$$

Es claro que en el caso de un rayo, al cambiar el punto q vamos a representar un rayo diferente. Mientras que podemos cambiar el vector v por un escalar positivo y seguimos representando el mismo rayo. Si cambiamos el vector por un escalar negativo vamos a obtener el rayo opuesto.

Por convención en el modelo de cámara estenopeica, en un proyector la luz viaja en la dirección definida por el vector dirección, mientras que en una cámara viaja en sentido opuesto.

Veamos ahora, sin pérdida de generalidad, cómo se logra triangular un punto 3D a partir de su observación desde dos cámaras.

En la teoría los rayos formados por la proyección de un punto 3D P en dos planos de imagen interseca en dicho punto (Figura 2.9). Pero hay que tener en cuenta que el modelo de cámara estenopeica general no es exacto, y que al ser digital las imágenes se discretizan en píxeles. La discretización hace que los rayos se calculen a partir de los píxeles donde se observa P. Juntando esto con la inexactitud del modelo (por más mínima que sea), en la pr'actica los rayos calculados tienen nulas probabilidades de intersecarse.

Veamos entonces cómo se resuelve este problema. Consideremos las líneas L_1 y L_2:

$$L_1 = \{\ p = q_1 + \lambda_1 v_1\ :\ \lambda_1 \in \mathbb{R}\ \}$$
$$L_2 = \{\ p = q_2 + \lambda_2 v_2\ :\ \lambda_2 \in \mathbb{R}\ \}\ .$$

Primero contemplemos los casos especiales. Los vectores v_1 y v_2 pueden ser linealmente dependientes o independientes.

Los dos rayos son paralelos si los vectores v_1 y v_2 son linealmente dependientes. Si además $q_2 - q_1$ es un escalar de v_1 o v_2 entonces las líneas son idénticas. Es claro que si las líneas son paralelas pero no idénticas, no se intersecan.

Si v_1 y v_2 son linealmente independientes, puede que las líneas intersequen o no. Si intersecan, intersecan en un punto. Esto significa que existe un $\lambda 1$ y $\lambda 2$ tal que

$$q_1 + \lambda_1 v_1 = q_2 + \lambda_2 v_2. \tag{2.19}$$

Como dijimos en previamente, en la práctica las líneas generadas no intersecan. Es por esto que definimos la intersección aproximada de dos líneas como el punto más cercano a ambas líneas. Es decir, intersequen o no, definimos la intersección aproximada como un punto p que minimiza la suma del cuadrado de su distancias a ambas líneas $\phi(p, \lambda_1, \lambda_2) = \|q_1 + \lambda_1 v_1 - p\|^2 + \|q_2 + \lambda_2 v_2 - p\|^2$ (Figura 3.13). Asumimos que los vectores v_1 y v_2 son linealmente independientes así p es único.

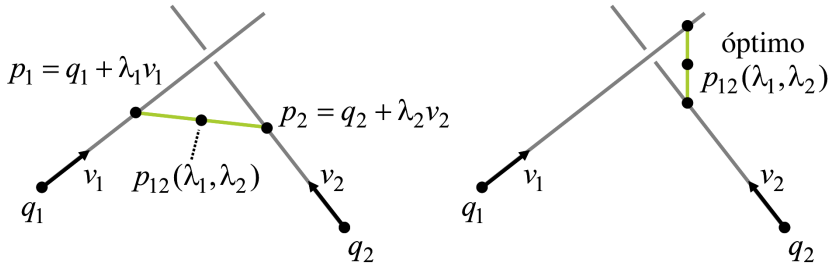


Fig. 2.11: Intersección aproximada de dos líneas/rayos.

Demostremos que esto es así y además veamos como calcular dicho punto. La función $\phi(p, \lambda_1, \lambda_2)$ es cuadrática semidefinida positiva de 5 variables: 3 coordenadas para el punto p y dos para los escalares λ_1 y λ_2.

Primero reducimos el problema a una minimización de una función cuadrática semidefinida positiva diferente, esta vez de solo dos variables λ_1 y λ_2. Sea $p_1 = q_1 + \lambda_1 v_1$ un punto en la línea L_1 y $p_2 = q_2 + \lambda_2 v_2$ uno en la línea L_2. Definimos el punto medio p_{12} del segmento que une p_1 con p_2 como:

$$p_{12} = p_1 + \frac{1}{2}(p_2 - p_1) = p_2 + \frac{1}{2}(p_1 - p_2)\ . \tag{2.20}$$

La condición necesaria para que el minimizador $(p, \lambda_1, \lambda_2)$ de ϕ es que las derivadas parciales de ϕ, con respecto a las 5 variables, sean 0 en el minimizador. En particular, las tres derivadas con respecto a las coordenadas de p deben anularse

$$\frac{\partial \phi}{\partial p} = (p - p_1) + (p - p_2) = 0. \tag{2.21}$$

o, equivalentemente, es necesario que $p = p_{12}$.

Es por esto que ahora el problema se reduce a minimizar la distancia cuadrada de un punto $p_1 \in L_1$ a un punto $p_2 \in L_2$. Osea, minimizar la función cuadrática semidefinida positiva de dos variables $\psi(\lambda_1, \lambda_2) = 2\phi(p_{12}, \lambda_1, \lambda_2) = \|(q_2 + \lambda_2 v_2) - (q_1 + \lambda_1 v_1)\|^2$.

Notemos que aún es condición necesaria que las derivadas parciales de ψ con respecto a λ_1 y λ_2 sean 0 en el mínimo:

$$\frac{\partial \psi}{\partial \lambda_1} = v_1^t(\lambda_1 v_1 - \lambda_2 v_2 + q_1 - q_2) = \lambda_1 \|v_1\|^2 - \lambda_2 v_1^t v_2 + v_1^t(q_1 - q_2) = 0 \tag{2.22}$$

$$\frac{\partial \psi}{\partial \lambda_2} = v_2^t(\lambda_2 v_2 - \lambda_1 v_1 + q_2 - q_1) = \lambda_2 \|v_2\|^2 - \lambda_2 v_2^t v_1 + v_2^t(q_2 - q_1) = 0 \tag{2.23}$$

Esto nos da dos ecuaciones lineales en λ_1 y λ_2 que se pueden expresar en forma de matriz como:

$$\begin{pmatrix} \|v_1\|^2 & -v_1^t v_2 \\ -v_2^t v_1 & \|v_2\|^2 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} v_1^t(q_2 - q_1) \\ v_2^t(q_1 - q_2) \end{pmatrix} \tag{2.24}$$

Por la independencia lineal de v_1 y v_2 la matriz izquierda es no-singular. Es por eso que la única solución del sistema lineal es:

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} \|v_1\|^2 & -v_1^t v_2 \\ -v_2^t v_1 & \|v_2\|^2 \end{pmatrix}^{-1} \begin{pmatrix} v_1^t(q_2 - q_1) \\ v_2^t(q_1 - q_2) \end{pmatrix} \tag{2.25}$$

Luego, la intersección aproximada p se puede obtener del valor de λ_1 o λ_2 obtenidos. Esta intersección aproximada es la que vamos a utilizar para triangular un punto 3D a partir de su proyección en varias imágenes.

2.2. Correspondencia por luz estructurada

Hasta ahora vimos los modelos y técnicas usadas para recuperar la ubicación de un punto 3D utilizando su proyección en varias imágenes. Lo que asumimos hasta ahora es que sabemos en qué lugar de las imágenes capturadas se observa dicho punto. Es decir, que conocemos el (U_L, V_L) y (U_R, V_R) para el ejemplo ilustrado en la Figura 2.9.

El problema de identificar qué puntos de una imagen corresponden a qué puntos de otra imagen se llama problema de correspondencia. Este problema fue y continúa siendo profundamente investigado en el área de Visión Computacional. Como hablamos en la introducción, en la práctica se utilizan tanto técnicas pasivas que comparan características visuales de las imágenes buscando similitudes, como técnicas activas que emiten distintas señales sobre la escena observada que luego pueden ser identificadas.

En esta tesis utilizamos una técnica activa basada en luz estructurada para resolver el problema de correspondencia. La luz se va a emitir sobre la escena utilizando un proyector, y se va a capturar la escena resultante con una o varias cámaras (Figura 2.12).

2.2.1. Luz estructurada

La luz estructurada consiste en patrones de luz específicamente diseñados para ser identificados en una imagen. Cada clase de patrón tiene un algoritmo de decodificación específico que va a permitir etiquetar zonas de la imagen capturada correspondientes a zonas del patrón. Un ejemplo básico sería proyectar un color único por píxel del proyector (Figura 2.13). En teoría si al capturar con dos cámaras una escena iluminada por tal patrón se logra identifica el color exacto de un píxel del proyector en ambas imágenes, sabríamos que corresponde al mismo punto.

En la práctica, un patrón como el descripto recién no es fácil de identificar en las imágenes, en especial si se busca hacerlo con precisión. Ya que para un proyector de

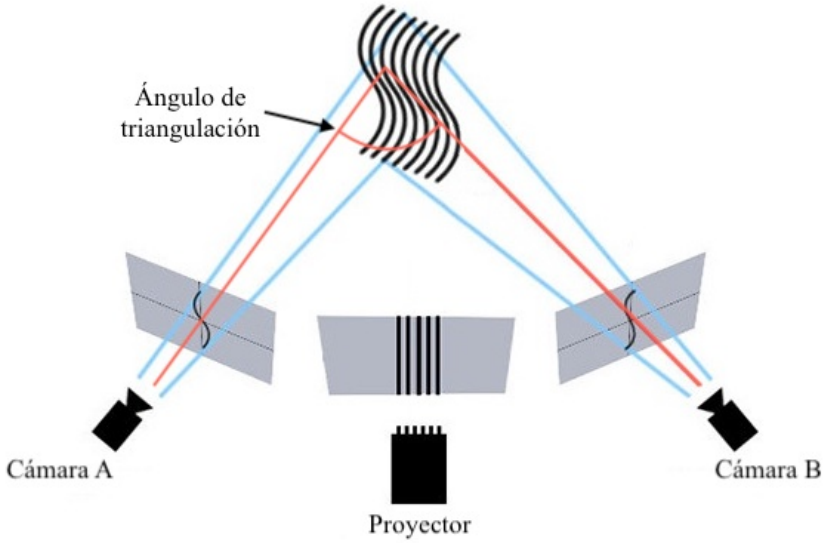


Fig. 2.12: Escaneo 3D utilizando luz estructurada.

800x600 habría que generar 480.000 colores diferentes, los cuales deberían ser identificables unívocamente en las imágenes. No solo las superficies que reflejan dicho patrón alteran el color, sino que los códigos de colores son lo suficientemente cercanos como para obtener un error significante en la decodificación.

Es por esto que en esta tesis utilizamos una clase de patrón temporal de codificación Gray, el cual es ampliamente usado en la práctica. Se dice que el patrón es temporal ya que no se proyecta un solo patrón sobre la escena, sino varios patrones uno después de otro, capturando cada resultado, y al decodificar se utiliza la información disponible en todas las imágenes.

El código Gray (Figura 2.14), también llamado código binario reflejado, es un reordenamiento del código binario clásico, tal que la representación de dos números decimales sucesivos difiera sólo en un bit. Esto logra que cada bit de la codificación contenga la misma medida de información. La traducción entre una palabra G de n $bits$ en código Gray y su representación B en binario es muy simple:

$$B_n = G_n$$
$$B_i = G_i \oplus G_{i+1} \ \forall i \in [1, n)$$

Siendo $\oplus$ la operación lógica XOR.

Los patrones temporales de luz estructurada de codificación Gray consisten en proyectar en cada píxel del patrón su número de columna seguido de su número de fila (Figura 2.14). Para un proyector de 800x600 vamos a necesitar por un lado codificar los números del 0 al 799 para las columnas, y del 0 al 599 para las filas. En ambos casos necesitamos $10bits$ ya que $2^{10} = 1024$. Esto significa que para lograr codificar las columnas y las filas necesitamos proyectar 10 patrones para cada bit de las columnas seguido de 10 patrones para cada bit de las filas.

Decodificar el número de columna y fila de un píxel consistirá en determinar su valor en cada imagen. Para un píxel se determina si en cada imagen observa un 1 o 0 y se

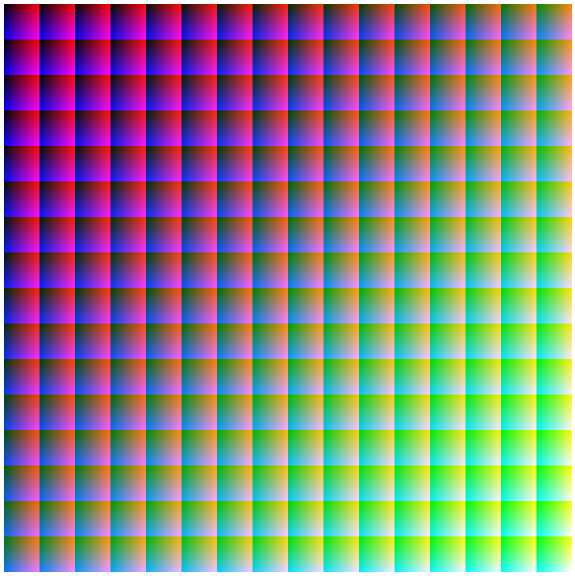


Fig. 2.13: Patrón de luz estructurada de codificación directa.

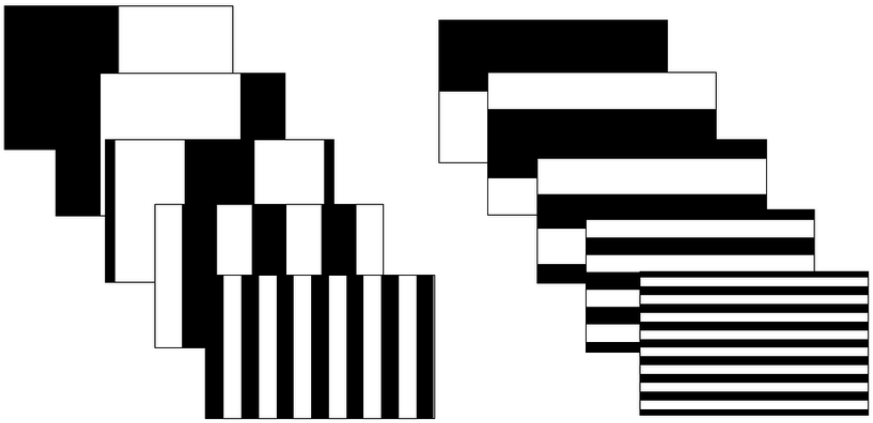


Fig. 2.14: Patrón de luz estructurada temporal de codificación Gray.

anexa cada dígito recuperado para luego convertir el código obtenido a su valor decimal. Decidir si un píxel de una imagen corresponde a un 0 (apagado) o un 1 (encendido) es el problema principal a resolver. Una técnica usada con mucho éxito en la práctica es la elaborada en [XA07] llamada *clasificación de píxel robusta*. Esta técnica modela no sólo la luz emitida por el proyector sino que también la luz ambiental que afecta a la escena. Estas son llamadas iluminación directa e iluminación global, respectivamente. Modelar ambas permite obtener una decodificación robusta de los datos. Para lograr esto no solo se proyectan los patrones mencionados previamente, sino que también se proyecta el negativo de cada uno. Siguiendo el ejemplo anterior, se proyectan 10 patrones para codificar la columna, 10 negativos de estos patrones, 10 para codificar las filas y sus 10 negativos, en total 40.

Con estos patrones la clasificación de píxel robusta propone las siguientes reglas para decidir si un píxel está encendido o apagado en una imagen capturada del patrón:

$$
\begin{aligned}
d < m &\rightarrow incierto \\
d > i_{total} \wedge p > \overline{p} &\rightarrow encendido \\
d > i_{total} \wedge p < \overline{p} &\rightarrow apagado \\
p < d \wedge \overline{p} > i_{total} &\rightarrow apagado \\
p > i_{total} \wedge \overline{p} < d &\rightarrow encendido \\
caso\ contrario &\rightarrow incierto
\end{aligned}
$$

Donde d es la incidencia de luz directa calculada para el píxel, i_{total} es la incidencia global de luz calculada para el píxel, p es la intensidad de luz en el píxel en la imagen capturada del patrón, $\overline{p}$ es la intensidad de luz en el píxel en la imagen capturada del patrón en negativo y m es un límite de luz mínima.

2.2.2. Correspondencia

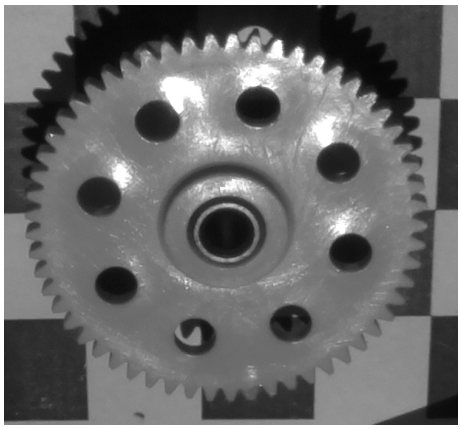


Fig. 2.15: Engranaje utilizado para ejemplos de captura.

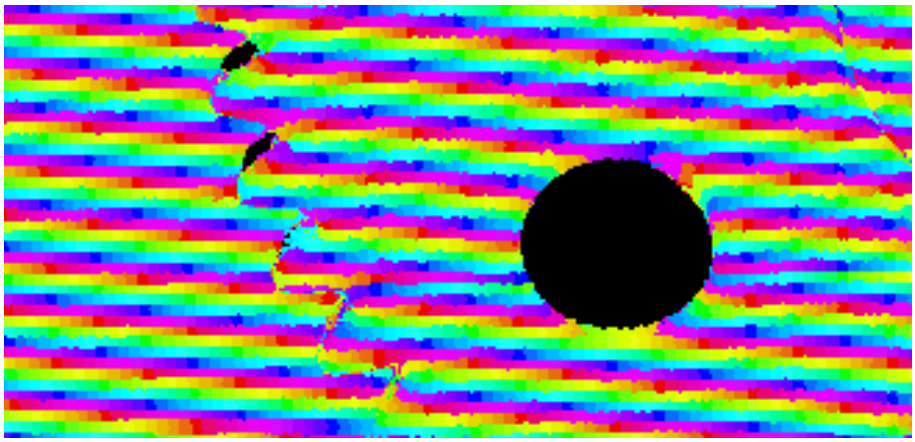


Fig. 2.16: Decodificación del patrón proyectado sobre un engranaje (Figura 2.15). Los colores se usan de forma cíclica para visualizar los píxel del proyector decodificados.

Los patrones de luz estructurada vistos en la subsección anterior nos permiten saber qué píxeles del plano imagen de una cámara corresponden con qué píxel del proyector. En el caso de utilizar varias cámaras nos permite determinar que partes de la escena son observadas por ambas cámaras, al decodificar el mismo valor en ambas imágenes.

Como vimos en el modelo de cámara estenopeica, el píxel de una cámara o un proyector no es un punto sino que tiene un tamaño. Esto se tiene que considerar a la hora de usar las correspondencias para triangular, ya que en la práctica las cámaras tienen más resolución que los proyectores, por lo que se suele decodificar el mismo valor en varios píxeles de la cámara (ver Figura 2.16).

Cada implementación resuelve este problema de forma diferente, por ejemplo eligiendo un representante para cada grupo o interpolando bilinealmente.

2.3. Trabajando con nubes de puntos

Lo explicado hasta ahora nos permite recuperar puntos 3D dado un sistema de cámaras y al menos un proyector calibrados a los modelos vistos.

Los puntos 3D generados en una captura se agrupan en lo que se llama una "nube de puntos". En esta sección veremos cómo se trabaja con estos puntos para lograr una representación fiel y útil de los objetos capturados. En especial veremos cómo se combinan varias nubes de puntos y cómo se generan superficies a partir ellas.

2.3.1. *Iterative Closest Point* (ICP)

Los puntos recuperados por triangulación en una misma captura se agrupan en una nube de puntos (*point cloud*). Naturalmente al querer recuperar puntos de un objeto en toda su periferia se necesitan realizar múltiples capturas. Cada captura observando al objeto desde diferentes ángulos. Cada captura resulta en una nube de puntos aislada. Es por esto que se necesitan técnicas para unir las nubes de puntos de forma controlada, tal que la agregación represente fielmente al objeto capturado.

Para agregar una nube de puntos N_B a otra nube de puntos N_A se aplica una rotación R y translación t a todos los puntos de la nube para luego unirlos en una nube de puntos N_{AB}. Esta rotación y translación se puede conocer o estimar.

Para conocer dicha transformación es necesario saber cómo se movió el objeto o el sistema de captura entre capturas. Una forma popular de lograrlo es colocando el objeto sobre una plataforma giratoria de precisión. Rotando dicha plataforma en ángulos conocidos entre capturas permite generar transformaciones acorde, que se aplican a la nube de puntos generada.

Aún utilizando hardware que mueve el objeto escaneado, si queremos capturar toda la superficie del objeto vamos a necesitar en algún momento reposicionarlo de forma manual, por ejemplo para capturar el punto de apoyo del objeto en la plataforma. En este caso, y en casos que no se utilice una plataforma móvil, desconocemos el movimiento que realizó el objeto de una captura a otra. Es por esto que existen técnicas que permiten estimar de forma robusta y precisa dicha transformación.

La técnica más utilizada en la práctica se conoce como Iterative Closest Point (ICP) [BM92]. Al igual que antes busca agregar una nube de puntos N_B a otra N_A.

La técnica consiste en aplicar los siguientes pasos:

1. Se empareja cada punto de la nube N_B con su más cercano de la nube N_A.

2. Se calcula la rotación y translación que minimice la suma de la distancia media cuadrática entre todos los pares.

3. Se aplica dicha transformación a la nube N_B.

4. Se evalúa un criterio de parada, en caso negativo se repiten los pasos.

El criterio de parada suele tener en cuenta la mejora lograda en cada iteración y la cantidad total de iteraciones. Existen numerosas implementaciones alternativas con detección de outliers y que trabajan con superficies en vez de puntos.

2.3.2. Reconstrucción de superficie

Una nube de puntos no constituye una superficie en sí como en su definición matemática. Esto presenta un gran problema ya que muchas aplicaciones que se le suele dar al escaneo de un objeto necesitan conocer cuál es el interior y exterior del objeto.

Es por esto que en muchos casos se busca reconstruir la superficie del objeto a partir de la nube de puntos escaneada. La reconstrucción de superficies a partir de nubes de puntos es un campo de visión computacional ampliamente estudiado.

Las numerosas técnicas desarrolladas suelen dividirse entre las que buscan generar triángulos usando los puntos como vértices, y las intentan reconstruir la función implícita de la superficie del objeto. En esta tesis trabajaremos con una técnica que pertenece al último grupo, llamada reconstrucción de superficie por Poisson (Screened Poisson Surface Reconstruction) [KH13].

Una superficie implícita se define como un conjunto de nivel $S = \{p \in \mathbb{R}^3 : f(p) = \lambda\}$ de una función continua $f : V \to \mathbb{R}$ donde V es un subconjunto abierto en 3D. La reconstrucción de superficie por Poisson es un método que reconstruye la superficie a partir de una nube de puntos orientados. Un punto orientado consiste de un punto 3D junto a un vector que apunta en dirección al exterior del objeto en dicho punto. A este vector lo llamamos normal del punto. En caso de usar una nube de puntos simple, es decir sin las normales, el método las estima.

En resumen el método reduce el problema a computar una función escalar donde el gradiente de dicha función sea el más cercano a un campo de vectores generado a partir de las normales de los puntos. La reconstrucción de superficie por Poisson es utilizada ampliamente en la práctica por ser robusta contra muestras ruidosas y por su capacidad de generar una superficie suave y cerrada (*watertight*).

2.4. Desempeño de un instrumento de medición

Un escáner 3D es un instrumento de medición que mide la posición de un objeto en el espacio 3D. Para evaluar y comparar cualquier instrumento de medición se necesitan conocer características que lo definan. Si bien hay características que tienen sentido solo para ciertos instrumentos, las más utilizadas a la hora de conocer el desempeño de un instrumento son:

1. Exactitud: cuan cercano es el valor obtenido al valor real.

2. Precisión: cuanto varia el valor obtenido al realizar una misma medición múltiples veces.

3. Resolución: el tamaño del cambio mínimo que el instrumento es capaz de medir.

4. Rango: Los mínimos y máximos valores que el instrumento es capaz de medir.

5. Tiempo de respuesta: cuánto tarda el instrumento en capturar y reportar la medición.

Para una definición mas detallada de estos y otros conceptos de metrología se puede acudir al *Vocabulario internacional de metrología* [DB09] desarrollado por el *Joint Committee for Guides in Metrology*.

La eficiencia de un instrumento de medición está dada por la exactitud y precisión del mismo (Figura 2.18).

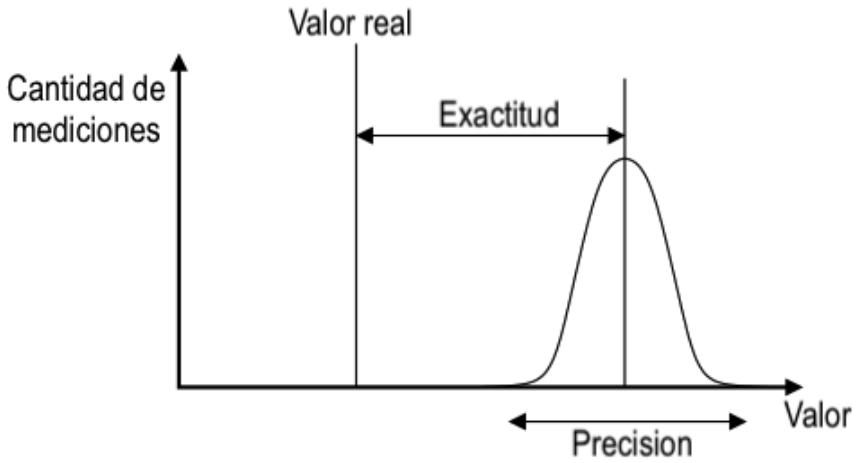


Fig. 2.17: Diferencia entre precisión y exactitud.

La precisión se mide utilizando el desvío estándar de la distribución de muchas mediciones como se observa en la Figura 2.18. Si bien hay consenso en como se mide la precisión, la exactitud tiene varias formas de medirse.

En el caso de esta tesis, la exactitud se mide utilizando la regla **x-sigma** siendo σ (sigma) el desvió estándar de la distribución de distancias de los valores medidos al valor real de referencia:

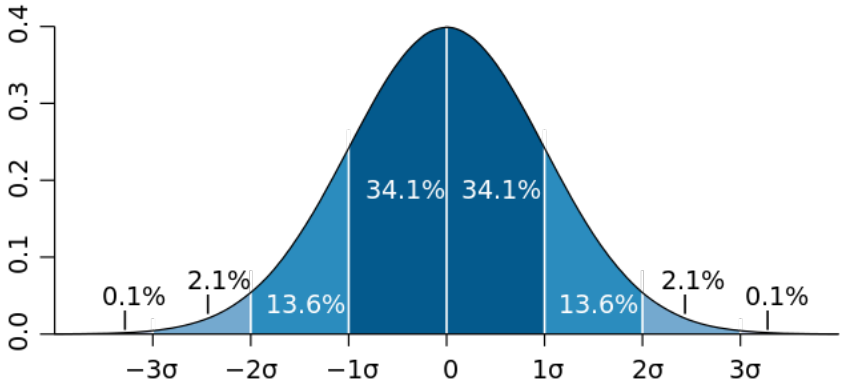


Fig. 2.18: Descripción gráfica de la regla x-sigma. Se puede observar a su vez la regla 68–95–99.7 que muestra la cantidad de muestras que entran en cada banda de sigma.

Dependiendo del instrumento de medición, la exactitud y precisión puede variar dependiendo el escenario en el que se lo use. Por ejemplo, en el caso de un instrumento que use cámaras con un modelo proyectivo, la imagen que capturan las cámaras es perspectiva. Esto causa que la exactitud, precisión y resolución dependan de la distancia del sistema de captura al punto medido.

3. DESARROLLO

En este capítulo vamos a recorrer de forma detallada el desarrollo completo del sistema de escaneo 3D. Vamos a explicar su diseño y su construcción, abarcando el hardware utilizado, su configuración y el software en su totalidad.

Como mencionamos previamente un objetivo de esta tesis es desarrollar un sistema de escaneo 3D integrado, que capture objetos 3D con súper-resolución. Otro objetivo principal de esta tesis es que dicho sistema sea estrictamente de bajo costo. El sistema de escaneo 3D se puede ver en el siguiente diagrama (Figura 3.23).

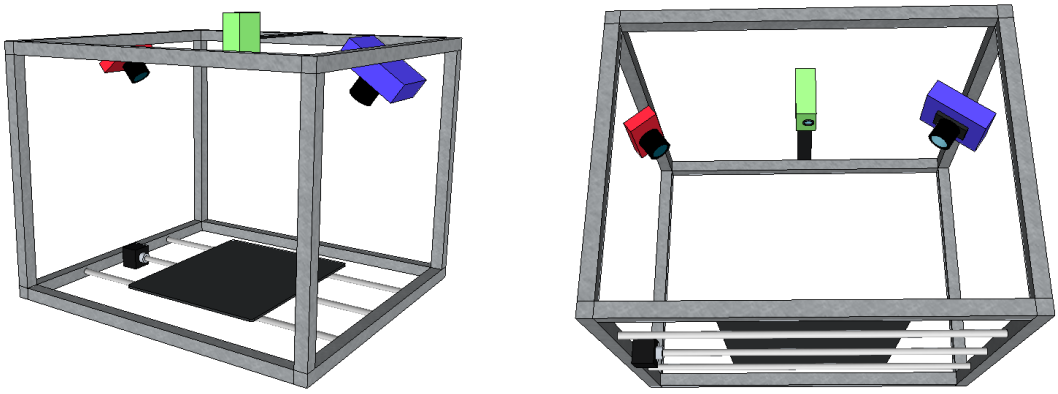


Fig. 3.1: Diseño del sistema de escaneo 3D.

El cual se compone físicamente de los siguientes componentes:

- Estructura: donde se acoplan los componentes.
- Computadora integrada: controla los dispositivos del sistema, los sincroniza para capturar un objeto, realiza el procesamiento de imágenes y cálculos de triangulación. A su vez ofrece una interfaz de usuario para utilizar el sistema (diagramada en azul).
- Cámaras: Para la captura de imágenes, una acoplada a la computadora, y la segunda a la estructura (diagramada en rojo).
- Proyector: para la proyección de patrones de luz estructurada, (diagramada en verde).
- Plataforma móvil: donde se coloca el objeto a escanear, y se mueve de forma precisa para capturar con súper-resolución (diagramada en negro).

Una vez calibrado el sistema, el proceso de escaneo 3D en este diseño consiste en colocar el objeto a escanear en la plataforma. Luego se procede a proyectar los patrones de luz estructurada con codificación Gray sobre el objeto y capturarlos de forma sincronizada con la cámara. El objeto luego es trasladado micrométricamente, para repetir el proceso de captura.

Luego, se procede a decodificar en cada imagen los patrones de luz estructurada, generando así las correspondencias entre las cámaras. A partir de las imágenes decodificadas se generan los rayos de las cámaras para cada correspondencia y se calcula su intersección aproximada para recuperar los puntos 3D que se encuentran en la superficie del objeto. Todo esto controlado a través de una aplicación web servida por el mismo escáner.

Para lograr súper-resolución, se desarrollaron dos técnicas. La primera es basada en hardware, y se logra utilizando la plataforma de movimiento lineal que, al mover el objeto escaneado micrometricamente, permite obtener una nube de puntos con mayor resolución. La segunda técnica es completamente algorítmica y está basada en tallado volumétrico (*space carving*), en la cual se usan los rayos de cámara correspondientes entre ambas cámaras para tallar un cubo 3D centrado en el área de trabajo. La segunda técnica se usa de forma complementaria a la primera técnica, es decir sobre los datos obtenidos por ésta.

3.1. Estructura del capítulo

En el desarrollo de esta tesis se lograron dos implementaciones funcionales de un escáner 3D, la primera es la versión V1 que, a pesar de funcionar, requiere de un mayor y más tedioso mantenimiento. La segunda versión V2 es la implementación finalmente elegida en esta tesis. La diferencia principal entre ambas versiones es la configuración de cámaras y proyector.

Si bien se mostrará el diseño y los principios de funcionamiento de la versión V1, es en la versión V2 en la que se enfocará este capítulo, ya que los fundamentos matemáticos en los que se basan ambos diseños son los mismos. Comenzaremos con una vista general seguido de un análisis detallado tanto del escáner desarrollado junto con las técnicas necesarias para obtener súper-resolución.

Finalmente describiremos la implementación de distintas partes del escáner de la siguiente forma:

1. Construcción física del escáner: diseño base, sus modificaciones y el agregado de hardware.
2. Hardware: proceso de inicialización e interfaces.
3. Software: implementación del proceso de calibración, captura de imágenes sincronizada con la proyección y decodificación de luz estructurada, problema de correspondencia, triangulación y la interfaz de usuario.

3.2. Vista General

El escáner 3D desarrollado (Figura 3.23) consiste de dos de cámaras de similar resolución y un pico-proyector (proyector portátil) orientados hacia una plataforma de movimiento lineal colocada en la base del escáner, donde se apoyará el objeto a escanear.

Al observar el objeto desde dos puntos de vista, y utilizando el proyector para resolver la correspondencia entre ambas observaciones, el escáner puede recuperar puntos 3D del objeto capturado utilizando triangulación.

Para resolver el problema de correspondencia el pico-proyector emite patrones de luz estructurada con codificación Gray sobre la escena a capturar. Al decodificarse en las imágenes capturadas, estos patrones permiten determinar qué puntos de la escena se observan en ambas imágenes capturadas. A su vez se utiliza la plataforma de movimiento

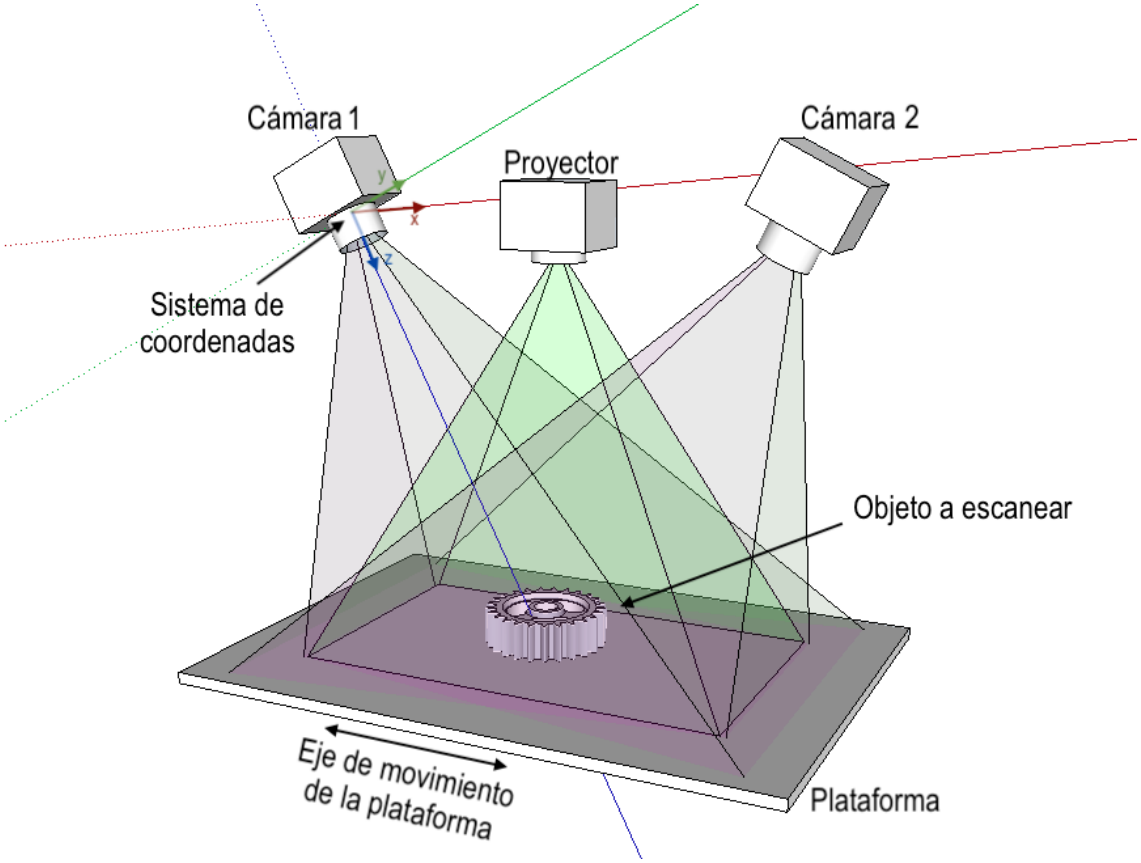


Fig. 3.2: Modelo matemático del escáner.

lineal para trasladar el objeto micrométricamente de forma controlada para luego repetir la captura. Si bien explicaremos más detalladamente la técnica luego, esto permite aumentar la resolución de sistema de captura, es decir, logra súper-resolución.

El sistema es controlado por el usuario a través de una interfaz web servida por el mismo escáner. Esta interfaz le permite utilizar a un usuario poco experimentado el escáner de forma íntegra. Es decir que ofrece una interfaz amigable que abarca desde la inicialización y configuración del escáner hasta el escaneo de un objeto 3D.

El sistema es embebido, es decir, la captura y procesamiento se realizan en el escáner mismo. Para esto se utiliza una computadora ARM Raspberry Pi 3B (RPi). La RPi controla las cámaras, el proyector y la plataforma móvil, y se encarga de orquestar la captura de forma sincronizada con la proyección y la traslación de la plataforma, para luego procesar los datos capturados y recuperar una nube de puntos 3D. La RPi al mismo tiempo sirve la interfaz web por la cual un usuario puede controlar el sistema.

El software del escáner está dividido en tres partes principales:

1. Motor del escáner 3D: se encarga de los modelos matemáticos y geométricos necesarios para el funcionamiento del escáner. A su vez se encarga del procesamiento de imágenes, decodificación de luz estructurada y generación de correspondencias. También se encarga del cálculo y triangulación de rayos de cámara.

2. Controlador de escáner: contempla la interfaz con los diferentes dispositivos que componen al escáner, y orquesta de forma sincronizada las capturas, proyecciones y

movimientos de plataforma.

3. Interfaz de usuario: ofrece un control intuitivo del escáner. Permite calibrar y configurar el escáner de forma asistida, y a su vez iniciar capturas en 3D y visualizar las nubes de puntos obtenidas. Se accede a través de un navegador web que pueda acceder a la red donde el escáner esté conectado.

Para el modelo ideal del escáner 3D propuesto, lo mejor en cuestión de velocidad es que la computadora integrada tenga acceso dedicado al *framebuffer* del proyector y de la cámara, y que los tres dispositivos estén sincronizados con un mismo reloj. Esto permitiría realizar la adquisición de imágenes de la forma más rápida posible. Si bien esto es posible requeriría un desarrollo de hardware dedicado que por el costo que implica, nos impediría mantener un presupuesto bajo. Es por esto que tomamos la decisión de utilizar dispositivos modulares de bajo costo que estén disponibles en el mercado.

3.3. Principios de funcionamiento del escáner

En esta sección vamos a explicar el modelo matemático del escáner diseñado (Figura 3.2), como se ajusta (o calibra) el modelo a una configuración específica de dos cámaras, un proyector, y una plataforma de movimiento lineal. También, explicaremos cómo a partir del modelo matemático se pueden recuperar puntos 3D junto con los algoritmos y la matemática detrás de ambas técnicas de súper-resolución desarrolladas.

3.3.1. Calibración del sistema

El proceso de calibración del sistema consiste en ajustar el modelo geométrico propuesto (Figura 3.2) a una configuración específica de dispositivos. Este proceso consiste de los siguientes pasos:

- Calibración intrínseca de cada cámara: para calcular los rayos de cámara que construye cada píxel de la imagen capturada con respecto al centro de proyección de la cámara.
- Calibración extrínseca de ambas cámaras: para modelar ambas cámaras en un único sistema de coordenadas.
- Calibración de plataforma de movimiento lineal: Para conocer el vector de movimiento de la plataforma y por lo tanto controlar la ubicación del objeto cuando sea necesario.

Para calcular la **calibración intrínseca** de las cámaras se implementó la técnica desarrollada por Zhang et al. en [Zha99]. Como explicamos en la Subsección 2.1.2 el proceso consiste en capturar y detectar un patrón de calibración en varias posiciones y orientaciones con respecto a la cámara. Este proceso permite ajustar el modelo de cámara estenopeica con distorsión a una cámara digital. El modelo ajustado va a permitir calcular el rayo que construye a cada píxel de la imagen capturada por dicha cámara en un sistema de coordenadas del mundo con origen en el centro de proyección de la misma.

Para la **calibración extrínseca** de las cámaras, es decir estimar la pose de una cámara con respecto a la otra, se implementó la técnica mencionada también en la Subsección 2.1.2

que consiste en capturar el mismo patrón de calibración utilizado antes desde ambas cámaras simultáneamente. Al calcular la pose del tablero con respecto a cada cámara, se puede calcular la pose de una cámara con respecto a la otra (ver la Ecuación 2.16). Esto permite modelar ambas cámaras en un mismo sistema de coordenadas. Sin pérdida de generalidad, se decidió utilizar como sistema de coordenadas del escáner al sistema de coordenadas en milímetros de una de las cámaras (llamada "cámara de referencia" o C_R), descripto en la Figura 2.3. A partir de ahora nos vamos a referir a este sistema de coordenadas utilizado como SC_W.

Por último, la **calibración de plataforma** consiste en estimar la dirección de movimiento de la plataforma en el sistema de coordenadas SC_W. Para ello la técnica de calibración desarrollada consiste en apoyar el tablero de calibración sobre la plataforma tal que sea observado por la cámara C_R. Luego se procede a capturar imágenes del tablero utilizando C_R en distintas posiciones $(pp_0, ..., pp_n)$ de la plataforma. Para cada imagen capturada se calcula la posición del tablero con respecto C_R. Para lograr esto se estima la pose del tablero con respecto a la C_R de la misma manera que se hizo en las etapas de calibración intrínseca y extrínseca. Notar que al estimar la pose del tablero con respecto a una cámara, el tablero se modela como un plano en el espacio, con origen en la primera esquina interna del tablero (ver la Figura 2.7), y la traslación calculada es efectivamente la coordenada de dicha esquina en el sistema de coordenadas de la cámara. Por lo tanto al estimar la pose del tablero con respecto a C_R para cada imagen capturada, se obtienen las posiciones pt_i de dicha esquina en para cada posición pp_i de la plataforma (Figura 3.3).

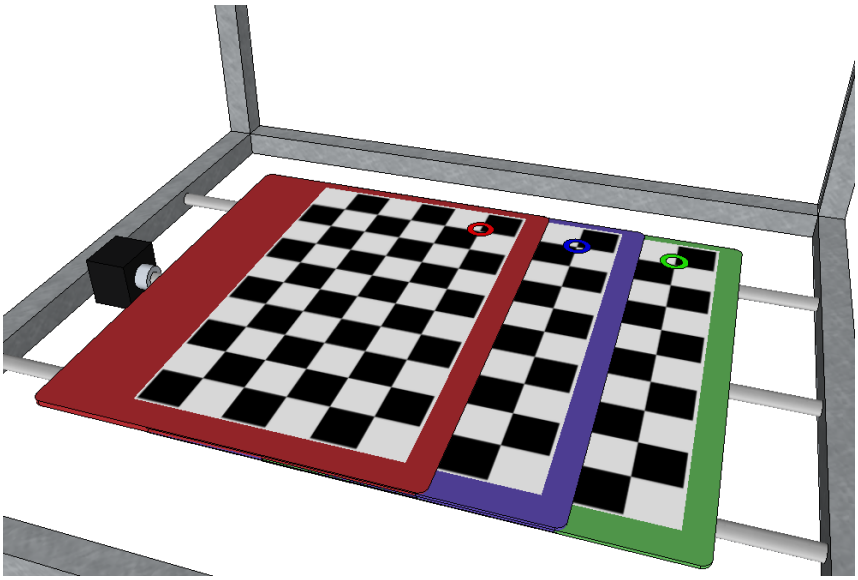


Fig. 3.3: El tablero capturado en distintas posiciones de la plataforma, cada color simboliza a la plataforma en una posición diferente. Los círculos marcan la posición de la primera esquina del tablero en cada posición de la plataforma.

Como sabemos que el movimiento de la plataforma es lineal, ajustar una línea l_p (Ecuación 3.1) a los pt_i permite recuperar el eje de movimiento de la plataforma (Figura 3.4).

$$l_p = \{\ p = q_p + \lambda_p v_p\ :\ \lambda_p \in \mathbb{R}\ \} \tag{3.1}$$

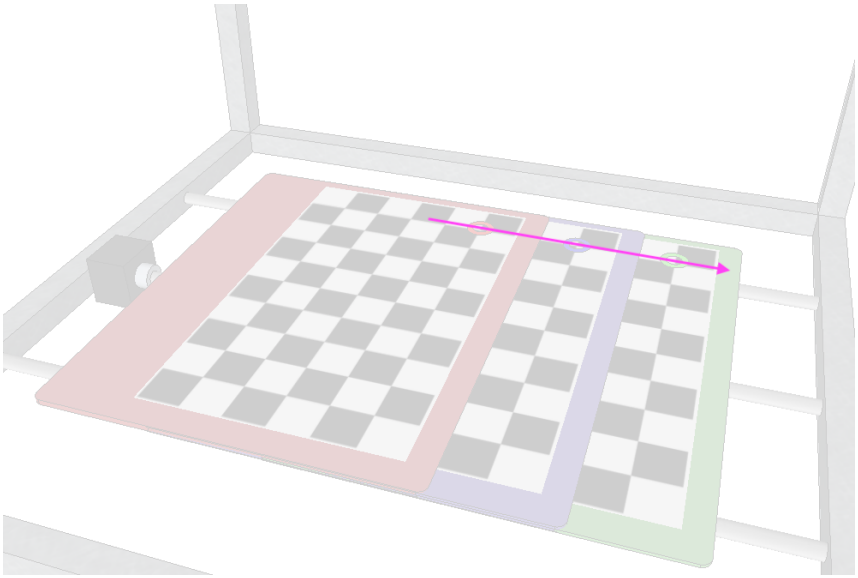


Fig. 3.4: Vector de movimiento ajustado al movimiento de la plataforma.

La línea que mejor ajusta a los puntos pt_i se puede obtener con un **análisis de componentes principales** (PCA) de la matriz:

$$P = \begin{pmatrix} pt_0^t \\ pt_1^t \\ \dots \\ pt_n^t \end{pmatrix} \tag{3.2}$$

Específicamente la primera componente principal va a permitir definir el vector dirección de la línea que buscamos, ya que la primera componente principal corresponde a la dirección de mayor varianza de los datos.

Una forma de definir la primer componente principal de la matriz P es la línea que su dirección es el autovector de la matriz de covarianza de P correspondiente al autovalor más grande.

Como PCA es sensible al centrado de los datos primero necesitamos centrar los puntos pt_i en el sistema de coordenadas, para ello primero se calcula la media de los puntos:

$$pt_{media} = \frac{\sum_{n=0}^{n} pt_i}{n} \tag{3.3}$$

y se le resta a cada punto:

$$P_{centrado} = \begin{pmatrix} pt_0^t - pt_{media}^t \\ pt_1^t - pt_{media}^t \\ \dots \\ pt_n^t - pt_{media}^t \end{pmatrix} \tag{3.4}$$

Luego para obtener los autovalores y autovectores de $P_{centrado}$ calculamos su **descomposición en valores singulares** (SVD):

$$P_{centrado} = U \Sigma V^t \tag{3.5}$$

La primera fila de V^t corresponde al autovector correspondiente al autovalor más grande de la matriz de covarianza A^tA de A. Por último, para calcular el vector dirección del movimiento de la plataforma $d_{plataforma}$ normalizamos dicho autovector y lo transponemos:

$$d^t_{plataforma} = \frac{V^t_0}{\|V^t_0\|} \tag{3.6}$$

La normalización provoca que la longitud de $d_{plataforma}$ sea de un milímetro. Veremos mas adelante como utilizamos este vector dirección para obtener súper-resolución.

Una observación muy importante a destacar es que se asume la linealidad del movimiento de la plataforma. Es decir, que el modelo propuesto no considera imperfecciones en el movimiento de la plataforma que puedan introducir un desvió al movimiento. Si bien en la práctica encontramos que esto no afecta los resultados (se verá más adelante en la experimentación) es importante el uso de una plataforma de movimiento con una construcción que logre un movimiento lo más lineal posible. Esto no es un requerimiento costoso de cumplir, ya que en la práctica no es costoso obtener ejes de movimiento rectos de alta precisión.

Una vez finalizado el proceso de calibración, ya estimamos todo el modelo necesario para el funcionamiento del escáner. En resumen calculamos:

- Modelos de cámara estenopeica con distorsión obtenidos en la calibración intrínseca: permiten calcular los rayos de cámara para cada píxel de las imágenes capturadas.
- Pose de la cámara secundaria con respecto a la cámara de referencia: permite llevar los rayos calculados por la cámara secundaria al sistema de coordenadas de la cámara de referencia y por lo tanto al sistema de coordenadas del sistema. Esto nos permite calcular la intersección aproximada entre el rayo de una cámara y uno de la otra.
- Vector dirección de movimiento de la plataforma: permite calcular el movimiento del objeto entre capturas, y por lo tanto nos permite combinar de forma precisa los puntos recuperados en ambas capturas.

3.3.2. Recuperación de puntos 3D

Como se vio en la Subsección 2.1.3, al haber calibrado las cámaras tanto intrínseca como extrínsecamente tenemos un modelo matemático que permite triangular objetos observados por ambas cámaras. Para lograrlo se coloca un objeto dentro del campo de vista y rango de foco de ambas cámaras y se captura con cada cámara.

Luego se procede a identificar los puntos en las imágenes correspondientes entre sí, y se generan los rayos de cámara para cada uno de los puntos usando el modelo de cámara estenopeica con distorsión. Luego se llevan los rayos pertenecientes a la cámara secundaria al sistema de coordenadas del escáner utilizando la rotación y translación calculados en la calibración extrínseca.

Una vez que están definidos todos los rayos en el mismo sistema de coordenadas se procede a calcular la intersección aproximada (Figura 3.13) entre rayos correspondientes. Las intersecciones aproximadas generan una nube de puntos 3D pertenecientes a la superficie del objeto.

La matemática para el cálculo de rayos de cámara se describió en la Subsección 2.1.1, específicamente en la Ecuación 2.14, mientras que la necesaria para el cálculo de la intersección aproximada se describió en la Ecuación 2.25. Esta matemática es relativamente simple una vez definidos los modelos. El problema principal surge a la hora de detectar los puntos correspondientes entre las imágenes capturadas por cada cámara.

3.3.3. Resolución problema de correspondencia

Como mencionamos previamente, en el escáner desarrollado este problema se resuelve en principio utilizando patrones de luz estructurada, específicamente patrones de luz con dimensión temporal de codificación Gray (Figura 2.14). Utilizando un proyector con resolución de H de alto por W de ancho, al proyectar estos patrones sobre la escena capturada se pueden decodificar e identificar unívocamente hasta $W * H$ valores correspondientes a cada píxel proyectado por el proyector.

Para la detección de los patrones en las imágenes capturadas el escáner desarrollado, implementa la *clasificación de píxel robusta* [XA07] y utilizando como parámetro $b = 0{,}2$ para la estimación de iluminación global y parámetro $m = 50$ para la clasificación de píxeles iluminados. Estos parámetros resultaron ser adecuados en nuestra configuración.

Como mencionamos en la Subsección 2.2.2, como generalmente en la misma clase de gama los proyectores digitales tienen menor resolución que las cámaras digitales, se suele decodificar un valor correspondiente a un píxel del proyector en varios píxeles de la cámara como se ve en la Figura 2.16, en la que cada bloque corresponde a un píxel del proyector decodificado. A partir de la decodificación de los valores en ambas imágenes capturadas (una por cada cámara) recae en el algoritmo de generación de correspondencias decidir qué píxeles se corresponden entre cada cámara.

El algoritmo de generación de correspondencia desarrollado en esta tesis selecciona un único píxel de la imagen como representante de cada valor decodificado exitosamente. Si volvemos a la Figura 2.16, esto significa que por cada color el algoritmo selecciona un píxel de la imagen como su representante. Esto permite que, si el valor es detectado en ambas imágenes, la correspondencia generada será entre los píxeles elegidos como sus respectivos representantes en cada imagen.

Para elegir al representante el algoritmo desarrollado elige el píxel más cercano al comienzo del bloque, siendo el bloque un píxel del proyector decodificado (en la Figura 2.16 sería un conjunto de píxeles de la imagen de un mismo color). Veamos a qué nos referimos con el comienzo del bloque.

Como vimos en la Subsección 2.2.1 el patrón de luz estructurada con codificación Gray codifica en cada píxel del proyector, primero su número de fila, y luego su número de columna. Al decodificar la codificación Gray cada píxel de la imagen, en caso de ser exitoso se obtiene el número de fila (Figura 3.5) y el número de columna del píxel del proyector observado (Figura 3.6).

El número de fila decodificado se encuentra en el rango $[0, H)$ mientras que el número de columna se encuentra en el rango $[0, W)$. Se considera como comienzo de una fila cuando se avanza un número de fila, y comienzo de una columna comienza cuando se avanza un número de columna.

Al no conocer la ubicación y orientación del proyector con respecto a las cámaras ni de qué forma se refleja el patrón en la escena capturada no se conoce una dirección específica dentro de la imagen en la que crezca el número de columna o el número de fila. En el ejemplo en el que se decodifican los patrones proyectados sobre un engranaje se puede

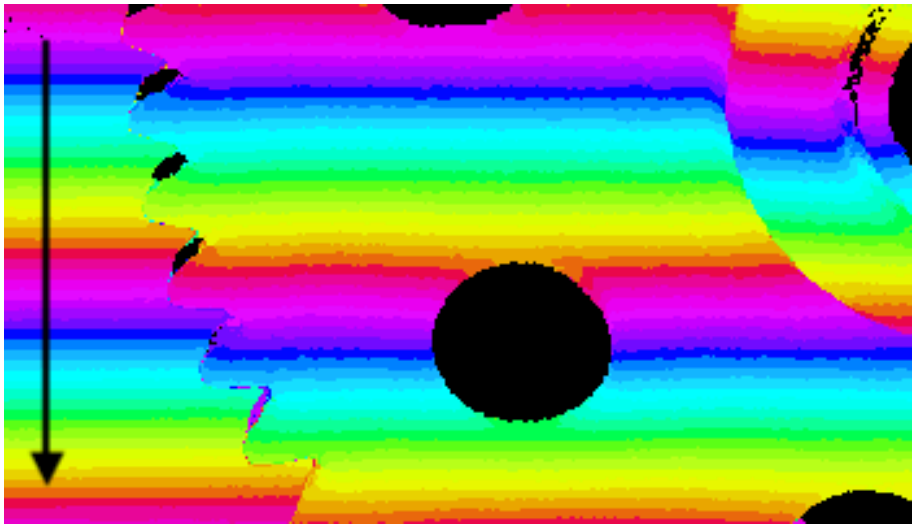


Fig. 3.5: Decodificación del patrón proyectado sobre un engranaje. Se muestra solo la decodificación de las filas del proyector. La flecha indica la dirección en el que el número de fila aumenta.

observar la dirección de crecimiento de fila decodificadas y columnas decodificadas con las flechas presentes en la Figura 3.5 y la Figura 3.6 respectivamente.

Teniendo en cuenta esto y utilizando el mismo ejemplo, los comienzos de filas y columnas definidos anteriormente se pueden observar en Figura 3.7 y Figura 3.9 respectivamente. Es decir, donde termina una fila o columna y comienza la siguiente.

Veamos ahora cómo se puede calcular el comienzo de cada fila del proyector. Al decodificar los patrones de luz se le asigna un valor de fila del proyector a cada píxel del plano imagen de la cámara ($[0, H)$), o el valor "indefinido" (u) en el caso que no se logre la decodificación del patrón. Los valores obtenidos se pueden organizar de forma similar a una imagen: como una matriz bidimensional $F \in D^{nxm}$ donde $D = \{u\} \cup [0, H)$ mientras que n y m son la resolución vertical y horizontal de la cámara respectivamente.

Luego para decidir si un píxel (i, j) de la cámara (es decir el ubicado en la fila i y la columna j) pertenece al comienzo de una fila de la decodificación se compara su valor de fila decodificado con el de los píxeles vecinos (ver Figura 3.8), si en algún caso el valor del píxel (i, j) es uno más que el del vecino, entonces el píxel (i, j) pertenece al comienzo de una fila de la decodificación. Formalmente esto sucede si:

$$
\begin{gathered}
f_{ij} = 1 + f_{i-1\ j-1} \vee f_{ij} = 1 + f_{i-1\ j} \\
\vee \\
f_{ij} = 1 + f_{i-1\ j+1} \vee f_{ij} = 1 + f_{i\ j-1} \\
\vee \\
f_{ij} = 1 + f_{i\ j+1} \vee f_{ij} = 1 + f_{i+1\ j-1} \\
\vee \\
f_{ij} = 1 + f_{i+1\ j} \vee f_{ij} = 1 + f_{i+1\ j+1}
\end{gathered}
$$

Análogamente, dada la matriz de decodificación de columna para cada píxel como $C \in D^{nxm}$ se puede decidir si el píxel (i, j) es simultáneamente el comienzo de una fila y una columna de la decodificación:

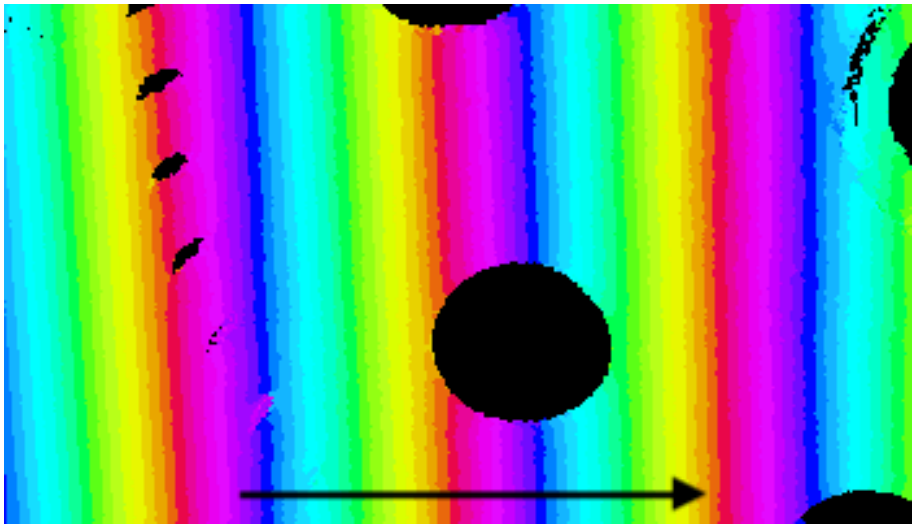


Fig. 3.6: Decodificación del patrón proyectado sobre un engranaje. Se muestra solo la decodificación de las columnas del proyector. La flecha indica la dirección en el que el número de columna aumenta.

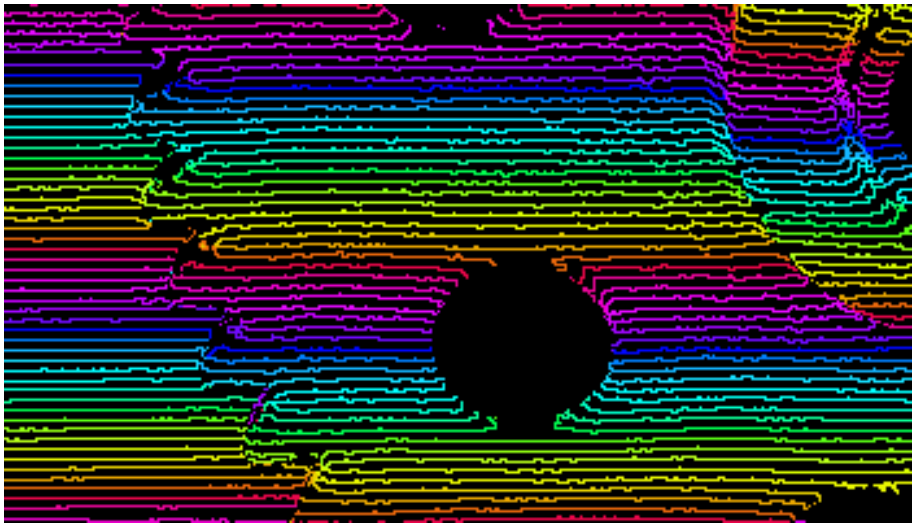


Fig. 3.7: Comienzo de fila del proyector en la decodificación.

$(x-1,y+1)$	$(x,y+1)$	$(x+1,y+1)$
$(x-1,y)$	(x,y)	$(x+1,y)$
$(x-1,y-1)$	$(x,y-1)$	$(x+1,y-1)$

Fig. 3.8: Vecinos de un píxel (x, y) en una imagen.

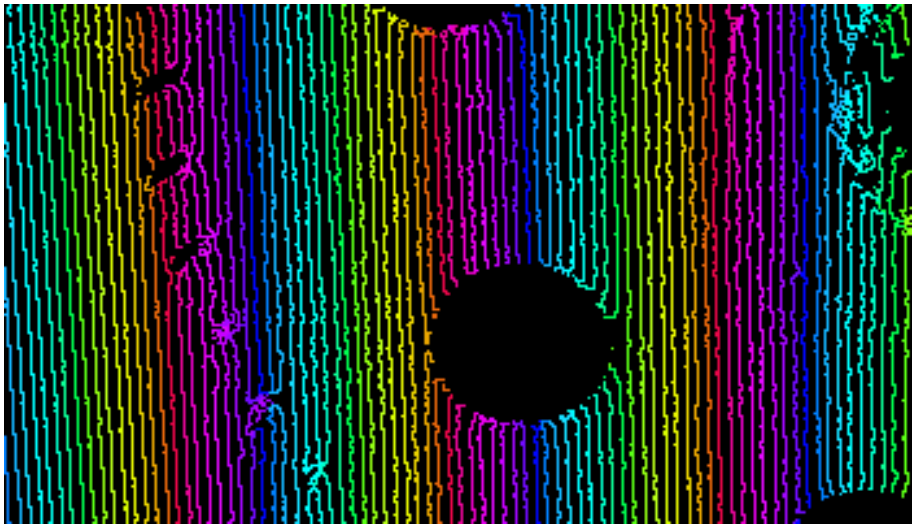


Fig. 3.9: Comienzo de columna del proyector en la decodificación.

$$
\begin{gathered}
isPixelOrigin(i,j) = \\
(f_{ij} = 1 + f_{i-1\ j-1} \wedge c_{ij} = 1 + c_{i-1\ j-1}) \\
\vee \\
(f_{ij} = 1 + f_{i-1\ j} \wedge c_{ij} = 1 + c_{i-1\ j}) \\
\vee \\
(f_{ij} = 1 + f_{i-1\ j+1} \wedge c_{ij} = 1 + c_{i-1\ j+1}) \\
\vee \\
(f_{ij} = 1 + f_{i\ j-1} \wedge c_{ij} = 1 + c_{i\ j-1}) \\
\vee \\
(f_{ij} = 1 + f_{i\ j+1} \wedge c_{ij} = 1 + c_{i\ j+1}) \\
\vee \\
(f_{ij} = 1 + f_{i+1\ j-1} \wedge c_{ij} = 1 + c_{i+1\ j-1}) \\
\vee \\
(f_{ij} = 1 + f_{i+1\ j} \wedge c_{ij} = 1 + c_{i+1\ j}) \\
\vee \\
(f_{ij} = 1 + f_{i+1\ j+1} \wedge c_{ij} = 1 + c_{i+1\ j+1})
\end{gathered}
$$

El algoritmo generador de correspondencias va a considerar sólo los píxeles de la cámara que cumplan tal condición. En la mayoría de los casos este filtro reduce a 1 la cantidad de píxeles de cámara correspondientes a un píxel específico del proyector (ver la Figura 3.10), pero hay casos en los que no, como se puede ver en la zona magnificada.

Esto representa un problema ya que se necesita un representante por píxel del proyector decodificado. Para lidiar con estos casos, se calcula el promedio de la posición de los píxeles de la cámara que comparten un mismo valor de decodificación del patrón de luz estructurada.

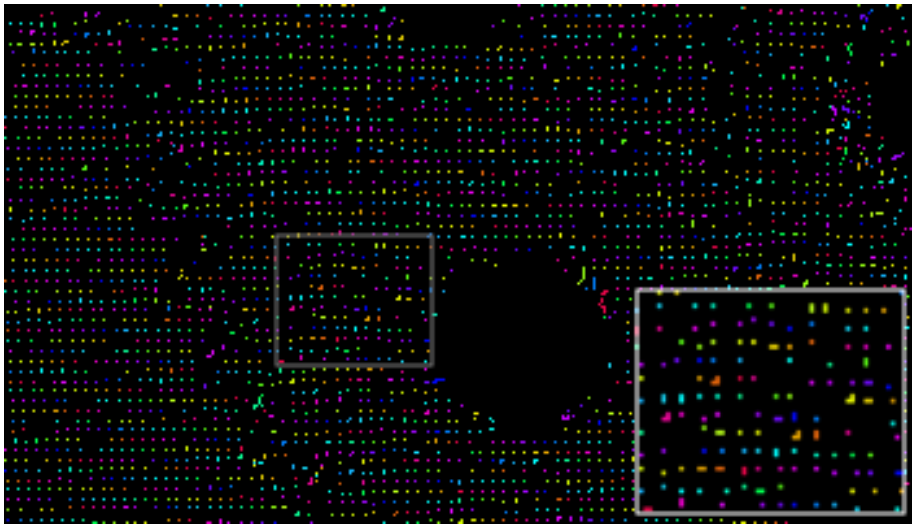


Fig. 3.10: Comienzo de píxel del proyector. En la zona magnificada Se puede observar que en algunos casos, el píxel de cámara detectado como comienzo de un píxel de proyector no es único.

Es decir que si el píxel (k, l) del proyector (con $k \in (0, H]$ y $l \in (0, W]$) fue decodificado exitosamente, su posición en el plano imagen es:

$$ProjectorPixelLocation(k, l) = Avg\left((i, j) \setminus f_{ij} = k \wedge c_{ij} = l \wedge isPixelOrigin(i, j)\right) \tag{3.7}$$

Siguiendo con el mismo ejemplo, el resultado de aplicar esta técnica se puede ver en la Figura 3.11.

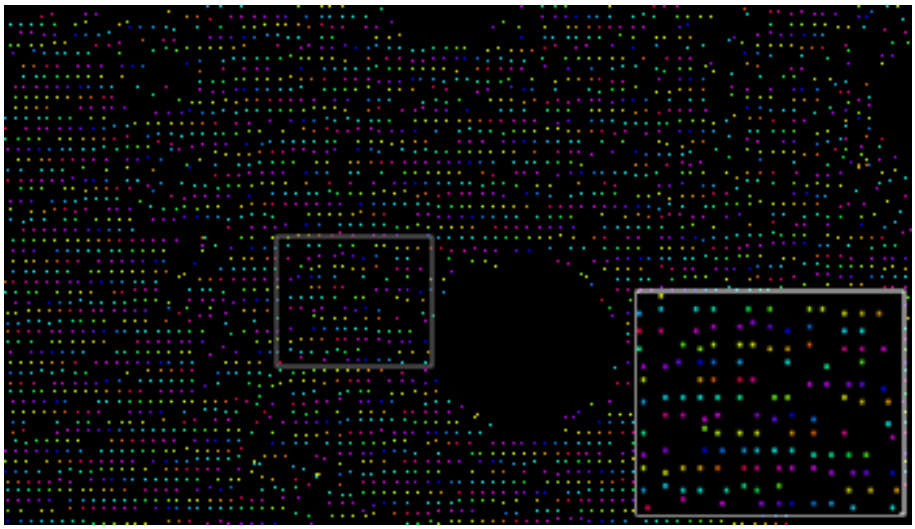


Fig. 3.11: Comienzo del pixel del proyector al calcularlos usando $ProjectorPixelLocation$.

Finalmente, el algoritmo de generación de correspondencias va a seleccionar los píxeles de proyector decodificados por ambas cámaras simultáneamente y generando una correspondencia por cada una. Esta correspondencia será entre sus respectivas posiciones en cada

plano de imagen. Estas son las correspondencias que serán utilizadas por el algoritmo de triangulación.

3.3.4. Súper-resolución utilizando la plataforma

La técnica de súper-resolución por hardware desarrollada tiene como objetivo reducir la limitación de resolución introducida por el proyector. Como vimos en la Subsección 3.3.3 al decodificar los patrones de luz estructurada en una imagen capturada, varios píxeles de la imagen corresponden al mismo píxel del proyector, y por eso elegimos en el plano imagen en representante de cada píxel del proyector. Todo algoritmo de correspondencias que no interpole datos va a generar un mapa de correspondencias que no cubre las imágenes capturas satisfactoriamente (si se ve la Figura 3.10 la zona negra son áreas no cubiertas). Esto empeora cuanto mayor sea la diferencia entre la resolución de las cámaras y la del proyector utilizado.

La técnica de súper-resolución desarrollada en esta tesis busca obtener datos reales en estos sectores no cubiertos por el mapa de correspondencias. Datos reales significa que no se generan a partir de los datos existentes, por ejemplo con interpolación, sino que se incrementa la cantidad de datos capturados. La técnica propuesta consiste en realizar varias capturas moviendo el objeto longitudes micrométricas conocidas utilizando la plataforma móvil. La longitud será tal que la nube de puntos obtenida logre una mayor cobertura de la superficie del objeto.

Como la falta de cobertura es causada por el tamaño del píxel del proyector, al ser proyectado sobre la escena, la longitud óptima de desplazamiento es menor a dicho tamaño. Para calcular el tamaño del píxel proyectado en el objeto es necesario saber el ángulo de visión del proyector y la distancia del proyector al área de trabajo del escáner. Ambas medidas se pueden observar en la Figura 3.12.

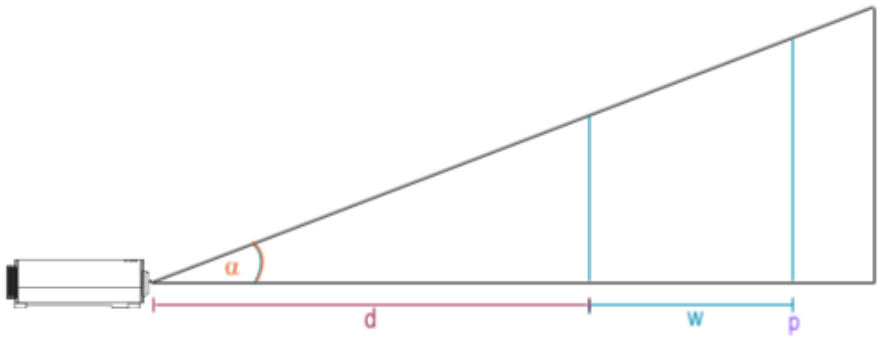


Fig. 3.12: Tamaño del campo de visión del proyector con respecto a la distancia de proyección. Vista lateral en el contexto del escáner desarrollado. α es el ángulo de visión del proyector, w el rango de trabajo del escáner (donde se ubican los objetos a escanear), d es la distancia del proyector al extremo del área de trabajo del escáner y p es la ubicación de la plataforma de movimiento (el otro extremo del área de trabajo).

En el caso del escáner diseñado en esta tesis, el proyector se coloca de tal manera que su plano imagen se encuentre paralelo a la plataforma de movimiento, y su eje y sea paralelo a la dirección de movimiento de la plataforma. Por esto, en un principio nos interesa sólo el tamaño vertical de los píxeles del proyector en el área de trabajo.

Para dar un ejemplo de un área de trabajo de un escáner 3D con respecto al proyector de luz estructurada, en el escáner diseñado en esta tesis dicha área comienza en la plataforma y alcanza unos $14cm$ sobre la plataforma. A su vez el proyector se encuentra a $24cm$ de la plataforma. Por lo tanto en este ejemplo $w = 10cm$ y $d = 14cm$.

Es claro que la altura del píxel del proyector varía dentro del área de trabajo: será más grande mientras más lejos se encuentre del proyector. En esta tesis la técnica de súper-resolución hará los cálculos basándose en el promedio de dicho tamaño dentro de toda el área de trabajo. El tamaño promedio del píxel del proyector dentro del área de trabajo se encuentra a $s = d + \frac{w}{2}$. Luego, como la proyección forma un triángulo rectángulo, para obtener la altura del píxel del proyector a distancia s del proyector primero se calcula el campo de visión vertical v del proyector a distancia s. Luego, se divide v por la resolución vertical del proyector H.

Para calcular v se utilizan reglas geométricas del triángulo rectángulo formado por la proyección del proyector. Para calcularlo a distancia s del proyector, tomamos a s como cateto adyacente al ángulo de visión, y a v como el opuesto del ángulo de visión del proyector α de. Luego como dicho triángulo rectángulo cumple con la Ecuación 3.8, basta con despejar v como se muestra en la Ecuación 3.9:

$$\tan(\alpha) = \frac{v}{s} \tag{3.8}$$

$$v = \tan(\alpha) * s \tag{3.9}$$

Luego el tamaño del píxel será $ppv = \frac{v}{H}$.

Volviendo a la técnica de súper-resolución, buscaremos mover la plataforma una distancia menor a ppv, tal que al realizar una nueva captura se recuperen puntos en zonas que previamente no se pudieron cubrir. La cobertura del área ínter píxel de proyector va a estar limitada por: la cantidad de capturas que se estén dispuesto a hacer y el movimiento mínimo que puede realizar la plataforma. Esto puede ser configurado con el único parámetro de configuración que tiene la técnica de súper-resolución: cantidad de particiones $cpart$ que indica en cuantas partes se va a dividir ppv, y por lo tanto definirá el movimiento realizado por la plataforma entre las capturas. Por ejemplo, en el caso del escáner implementado en la tesis, se eligió $cpart = 4$, por lo que se realizan 4 capturas con movimientos de plataforma de $\frac{ppv}{4}$ entre ellas.

Una vez realizadas las $cpart$ capturas, es necesario combinar las nubes de puntos obtenidas en cada paso en una nube de puntos única. Es en este momento en el que entra en juego el vector de movimiento de la plataforma ($d_{plataforma}$) estimado en el proceso de calibración. Siendo pc_i las nubes de puntos generadas por cada captura, con $i \in (0, ..., cpart - 1)$. La nube de puntos combinada pc es la siguiente:

$$pc = \bigcup_{i=0}^{cpart-1} \{p + t_i \mid p \in pc_i\} \tag{3.10}$$

$$t_i = \left(\frac{i * ppv}{cpart}\right) \; d_{plataforma} \tag{3.11}$$

Simplemente, se le aplica a todos los puntos de cada nube de puntos la traslación correspondiente de la plataforma para su respectiva captura y se combinan luego en una única nube de puntos.

A diferencia de capturar el objeto en diferentes posiciones y combinar los resultados usando ICP, esta técnica tiene dos principales ventajas. En primer lugar al estar modelando y trabajando sobre las deficiencias de la correspondencia por luz estructurada, agrega datos donde sabe que faltan, por lo es potencialmente más eficiente al no incluir datos redundantes. En segundo lugar, la combinación de nubes de puntos utilizando ICP es intrínsecamente una estimación, mientras que la técnica desarrollada modela y controla el movimiento del objeto en el mundo.

3.3.5. Escaneo volumétrico por *space carving*

Como vimos previamente el escáner 3D desarrollado calcula los rayos de las cámaras que se corresponden entre sí y los interseca de forma aproximada.

La técnica de escaneo volumétrico por *space carving* desarrollada en esta tesis consiste en utilizar dichos rayos de cámara para tallar un cubo 3D virtual de forma adaptativa, en busca de lograr súper-resolución.

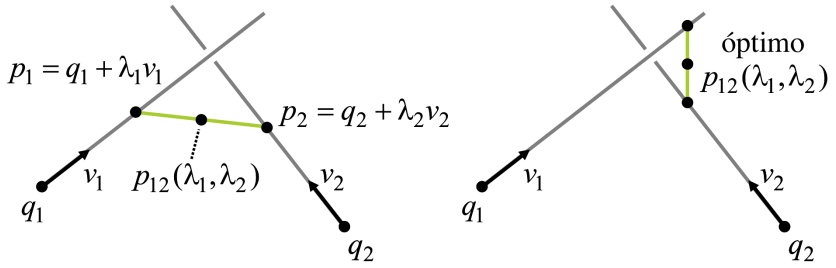


Fig. 3.13: Intersección aproximada de dos líneas/rayos.

Si observamos la Figura 3.13 q_1 y q_2 son los rayos correspondientes entre las cámaras. Para tallar el cubo se utilizan rayos con dirección opuesta a q_1 y q_2 y con origen en p_1 y p_2 respectivamente, los llamaremos $qo1$ y $qo2$.

El proceso de tallado va a dividir el cubo en dos: un área interna (el objeto escaneado) y un área externa (el área tallada). Ahora veremos su funcionamiento general, y luego lo definiremos formalmente.

El algoritmo comienza con un cubo alineado a los ejes del sistema de coordenadas o *Axis-aligned bounding box* ($AABB$) que se ubica aproximadamente en la zona de trabajo del escáner. Dicho cubo se talla de forma recursiva de la siguiente manera:

1. Se divide el cubo en 8 partes cortando por el centro de cada arista.

2. Cada sub-cubo -que también es un $AABB$- se clasifica en una de las siguientes clases: **interno** (si ningún rayo lo interseca), **borde** (si un rayo comienza dentro) o **externo** (si ninguno de los casos anteriores ocurre). La Figura 3.14 permite visualizar dicha clasificación.

3. Se aplica recursivamente a todos los sub-cubos clasificados como borde.

El algoritmo se detiene una vez que llega a un tamaño de lado de cubo mínimo definido en un parámetro $minBoxSideSize$. La técnica es adaptativa ya que aplica la recursión sólo a los sub-cubos borde, reduciendo el área de procesamiento de forma considerable. A

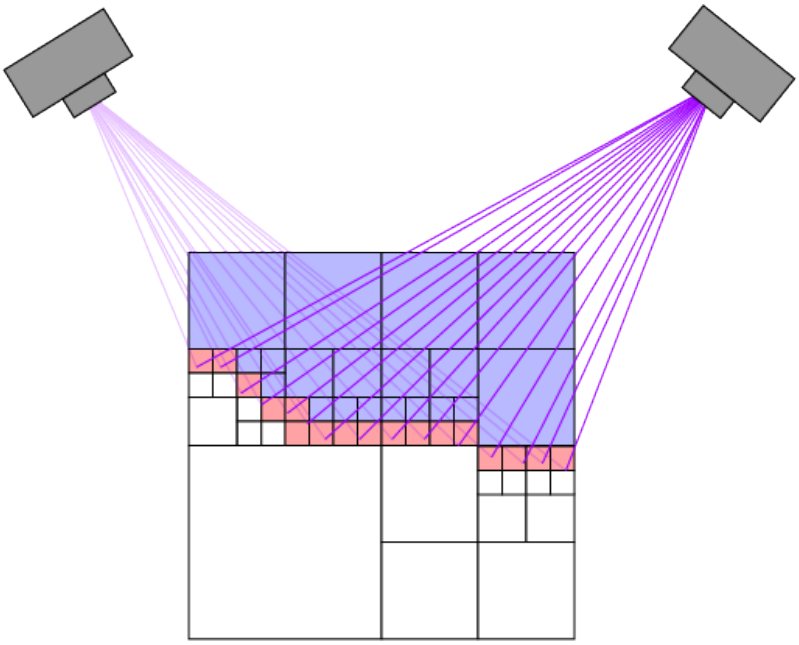


Fig. 3.14: Ejemplo del tallado propuesto en 2 dimensiones. Los rayos se visualizan en violeta y provienen de las cámaras (en gris). Los cubos "externos" se visualizan en azul, los "internos" en blanco y los "borde" en rojo.

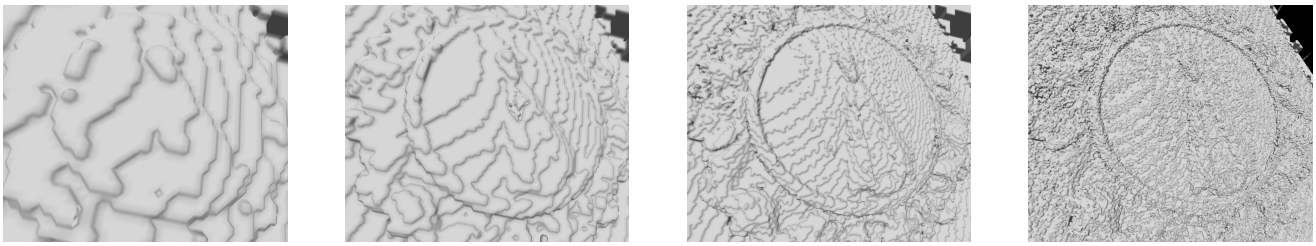


Fig. 3.15: Resultado del tallado volumétrico de una medalla a medida que se utiliza un $minBoxSideSize$ mas chico.

su vez la técnica permite tallar con granularidad arbitraria como se observa en la imagen Figura 3.15.

Veamos ahora paso por paso el funcionamiento del algoritmo. Definamos en un principio los rayos que vamos a utilizar. Como para el algoritmo es irrelevante que cámara se utilizó para generar cada rayo, vamos a nombrar a todos los rayos (es decir los $qo1$ y $qo2$ generados a partir de los q_1 y q_2 de cada correspondencia) como qo_i con $i \in [0\ ,\ 2 * \#C)$ siendo $\#C$ la cantidad de correspondencias recuperadas. Como vimos previamente, un rayo está compuesto por un origen y una dirección. El origen de qo_1 se llamará o_i y la dirección v_i

El primer paso del algoritmo consiste en elegir el $AABB$ a tallar. Un $AABB$ está definido por dos puntos $p0$ y $p1$, y en este caso particularmente deben formar un cubo, por lo que la distancia entre $p0$ y $p1$ debe ser la misma en todos los ejes del sistema. El algoritmo busca un cubo mínimo que cubra todos los orígenes o_i de los rayos qo_i. Para encontrar el tamaño mínimo $initialSize$ basta encontrar el valor mínimo y máximo que tienen los o_i en cada eje, y quedarse con el valor del eje que presente la mayor diferencia:

$$o_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \tag{3.12}$$

$$rangoX = max(x_i) - min(x_i) \tag{3.13}$$

$$rangoY = max(y_i) - min(y_i) \tag{3.14}$$

$$rangoZ = max(z_i) - min(z_i) \tag{3.15}$$

$$\forall i \in Qc,\ Qc = [0\ ,\ 2 * \#C) \tag{3.16}$$

$$initialSize = max(rangoX, rangoY, rangoZ) \tag{3.17}$$

Como el cubo de lado $initialSize$ que cubre todos los o_i suele no ser único la ubicación puede variar, aunque no afecta al funcionamiento del algoritmo. Una vez definido el bloque inicial se aplica el algorithm 1 al mismo:

Algorithm 1: CarveCube: Calculates sub-cubes and classifies them

```
input: cube: the cube to carve
       rays: an array containing all the rays to carve the cube with
       minBoxSideSize: minimum box size (stop condition)
subCubes ← CreateSubCubes(cube);
for subCube in subCubes do
    intersectedRays ← GetIntersectedRays(subCube, rays);
    containedRays ← GetRaysWithContainedOrigin(subCube, rays);
    if containedRays.lenght ≠ 0 then
        if cube.p1.x - cube.p0.x > minBoxSideSize then
            CarveCube(subCube,intersectedRays)
    else
        if intersectedRays.lenght = 0 then
            subCube.carved = True
        else
            subCube.carved = False
```

Dados un cubo y un conjunto de rayos `GetIntersectedRays` devuelve los rayos que intersecan dicho cubo. Para probar si un rayo (definido por un punto o y un vector dirección v) interseca un $AABB$ (definido por dos puntos $p0$ y $p1$) se utiliza una técnica desarrollada por Majercik et al. en [MCSM18] que utiliza el siguiente cálculo:

$$t0 = (p0 - o) \odot v$$
$$t1 = (p1 - o) \odot v$$
$$tmin = min(t0, t1)$$
$$tmax = max(t0, t1)$$
$$doesIntersect = maxComponent(tmin) <= minComponent(tmax)$$

Donde $doesIntersect$ indica si el $AABB$ y el rayo intersecan. Notar que $\odot$ es el producto Hadamard entre vectores, min y max también trabajan a nivel de componente de

vectores, es decir devuelven un vector, en cada posición i se encuentra el mínimo o máximo entre los elementos de la posición i en los vectores de entrada. Por último *maxComponent* y *minComponent* devuelven el máximo y mínimo componente del vector de entrada respectivamente.

Dados un cubo y un conjunto de rayos `GetRaysWithContainedOrigin` devuelve los rayos tales que su origen se encuentra dentro de los límites del cubo. Para probar si el origen o de un rayo se encuentra dentro de un $AABB$ (definido por dos puntos $p0$ y $p1$) se basta con probar que $p0 < o < p1$.

Para realizar esto de forma eficiente, y a su vez almacenar los resultados se utiliza una estructura de datos llamada *Octree*. Los nodos de un Octree representan un cubo en el espacio, y sus hijos (son siempre 8) corresponden a cada sub-cubo en los que se divide equitativamente el mismo.

Esta técnica de tallado desarrollada permite por ejemplo generar una grilla 3D (Figura 3.16) de muestreo de la superficie, cada vértice de la grilla clasificado como dentro o fuera de la superficie. Dicha estructura de muestreo es utilizada por varios algoritmos de reconstrucción de superficies famosos como *Marching Cubes* [LC87] o *Surface Nets* [Gib98].

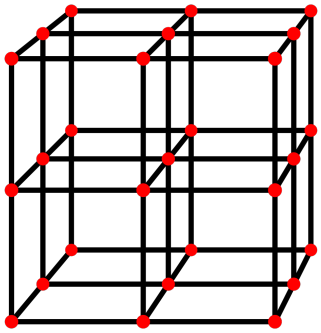


Fig. 3.16: Grilla 3D regular.

En esta tesis se implementaron dos algoritmos de reconstrucción de superficies que reciben como entrada el resultado del *space carving*. Primero se implementó el algoritmo de *Marching Cubes* sobre los cubos borde, y en segundo lugar se implementó un algoritmo simple que crea una nube de puntos a partir de los cubos clasificados como "borde". Este algoritmo genera un punto en el centro de cada cubo y luego aplica el algoritmo de *Screened Poisson Surface Reconstruction* sobre la nube de puntos generada.

3.4. Construcción

Para la estructura del escáner se tomó como base el diseño de un escáner 3D *laser slit*, realizado por Gabriel Taubin y Peter Walecki en la Universidad de Brown[1].

La estructura utiliza perfiles de aluminio de $25mm \ x \ 25mm$ de encaje en T (Figura 3.18a). Estos perfiles permiten un fácil ensamble y ofrecen buena flexibilidad de diseño.

El diseño se basa en utilizar un láser que forma una linea (plano láser en 3D) que al proyectarse sobre el objeto a escanear, el escáner calcule la intersección entre los rayos de cámara correspondientes a los píxeles donde se observa la luz del láser y el plano 3D que

[1] `http://mesh.brown.edu/taubin/`

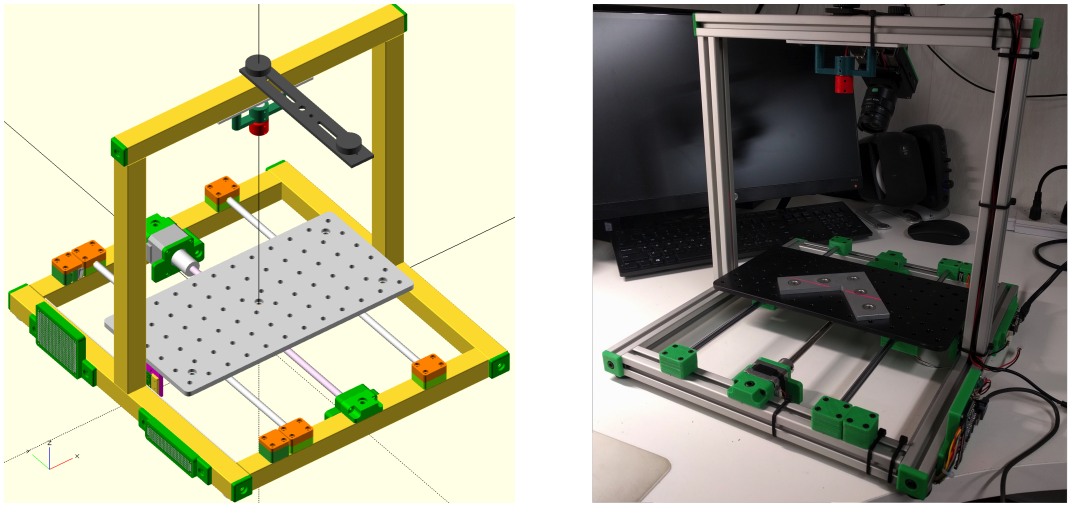


Fig. 3.17: Escáner 3D *laser slit* diseñado por Gabriel Taubin. Diseño en el que se basó el escáner desarrollado en esta tesis.

(a) Perfil con encaje en T (*t-slot*). *(b)* Lente de ajuste manual.

Fig. 3.18

el láser forma en el espacio. El láser contiene una lente cilíndrica que le permite producir la línea láser y está montado en un arco sobre la base del escáner.

Como en una captura el escáner recupera únicamente la parte del objeto iluminada por el láser, el diseño contempla una plataforma móvil que traslada al objeto a través del plano de luz para poder recuperar todo el objeto. Esta plataforma se mueve gracias a un motor paso a paso montado en la base del escáner. El eje del motor se acopla a una varilla de rosca trapezoidal a través de un acople flexible de aluminio. La plataforma se desliza sobre unos ejes de soporte y tracción gracias a un rulemán lineal que la acopla con la varilla roscada (Figura 3.20).

Para determinar la posición de la plataforma se utiliza un encóder magnético lineal junto a un sensor que lo decodifica (Figura 3.19). El encoder utilizado es un AS5311[2] de alta resolución que permite obtener un posicionamiento preciso de la plataforma (y por lo tanto del objeto). El mismo devuelve la posición de la plataforma con una resolución de

[2] `https://ams.com/as5311`

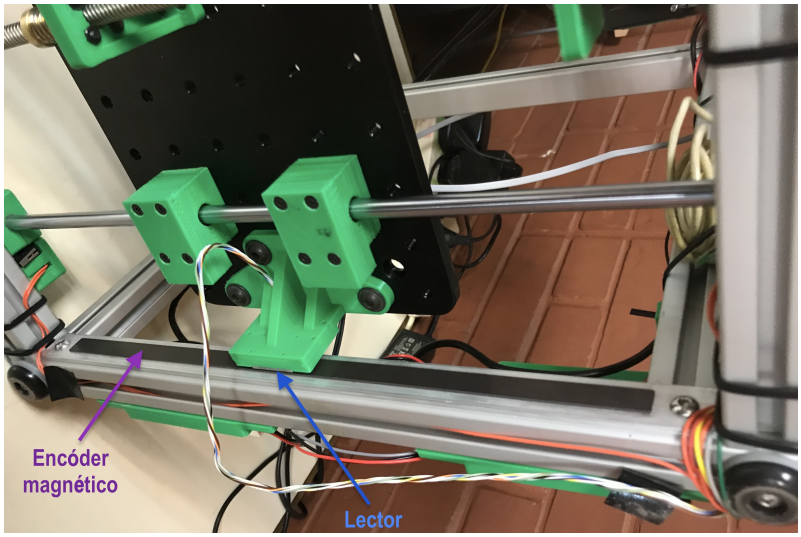


Fig. 3.19: Ubicación del encóder magnético lineal junto con su correspondiente cabezal lector.

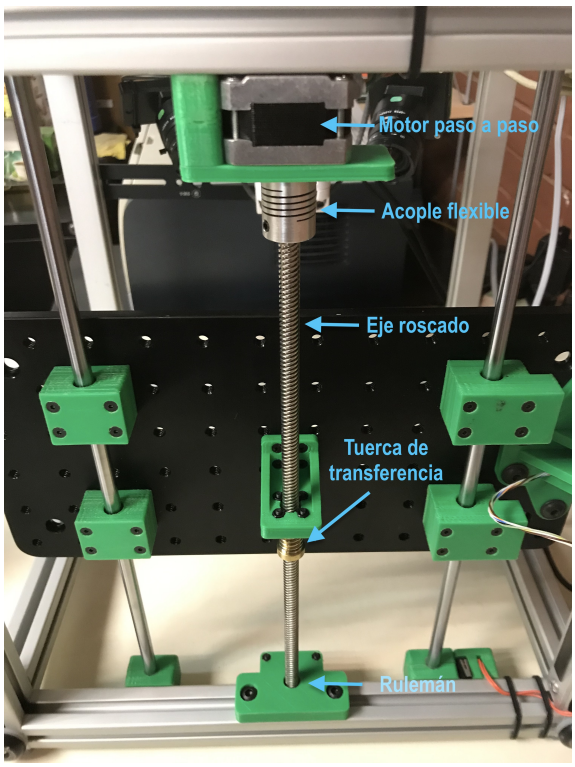


Fig. 3.20: Transmisión de la plataforma móvil.

488 *nm*.

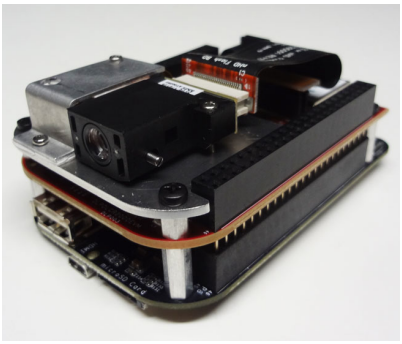
Para el control del motor de la plataforma se utiliza una placa Arduino Nano junto a un driver de motor paso a paso. La placa Arduino Nano a su vez lee la posición de la plataforma y controla el encendido del láser.

El diseño a su vez contempla una computadora reducida Raspberry Pi (RPi) 3B utilizada como procesador embebido. La RPi se encuentra conectada a la placa Arduino Nano

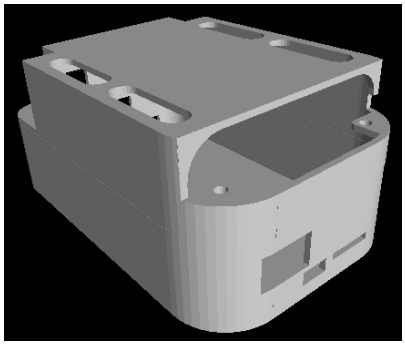
para controlar la plataforma y el haz de luz láser. A su vez la RPi se encuentra conectada a una cámara RPiCam V1 para la captura de imágenes de escaneo. La misma captura imágenes con una resolución de $2592x1944$ píxeles. La cámara RPiCam se adaptó para incluir una lente que permite el ajuste manual de foco, apertura y zoom Figura 3.18b. Todo el hardware es alimentado por una fuente de 12 voltios.

3.4.1. V1

Como mencionamos previamente la primera versión se basa en triangulación utilizando un par cámara/proyector. Para lograr esto se remplazó el módulo láser por un kit de desarrollo de Texas Instruments[3] Figura 3.21a.



(a) Proyector TI con BeagleBone-Black acoplada.



(b) Lente de ajuste manual.

Fig. 3.21

Este kit consiste del módulo de evaluación de un pico-proyector Texas Instruments LightCrafter 2000 acoplado a una computadora BeagleBone Black para control embebido del mismo. El pico-proyector tiene una resolución de $640x360$ píxeles.

La computadora BeagleBone utiliza Debian como sistema operativo y está conectada a través de sus pines de desarrollo al módulo de proyección.

Para poder acoplar el kit a la estructura del escáner se diseñó una carcasa a medida con guías que puede acoplarse al perfil T-slot con cierto grado de libertad (Figura 3.21b).

Esta carcasa se imprimió en 3D, se ensambló con el proyector, y se colocó en el arco superior de la estructura orientando el proyector hacia la plataforma con un ángulo de 90 grados Figura 3.22.

El proyector se conectó a la fuente para su alimentación, y se conectó a través de cable Ethernet a la RPi.

3.4.2. V2

Esta versión se basa en triangulación utilizando un par de cámaras, utilizando el proyector únicamente para resolver el problema de correspondencia. Además de esto, se buscó robustecer la estructura para no perder la calibración fácilmente al desplazar el escáner. Para esto en primer lugar se agregó un segundo arco de perfiles de aluminio. Luego se desplazó el arco original y se unieron los dos arcos con perfiles de aluminio de soporte.

[3] `https://www.ti.com/tool/DLPDLCR2000EVM`

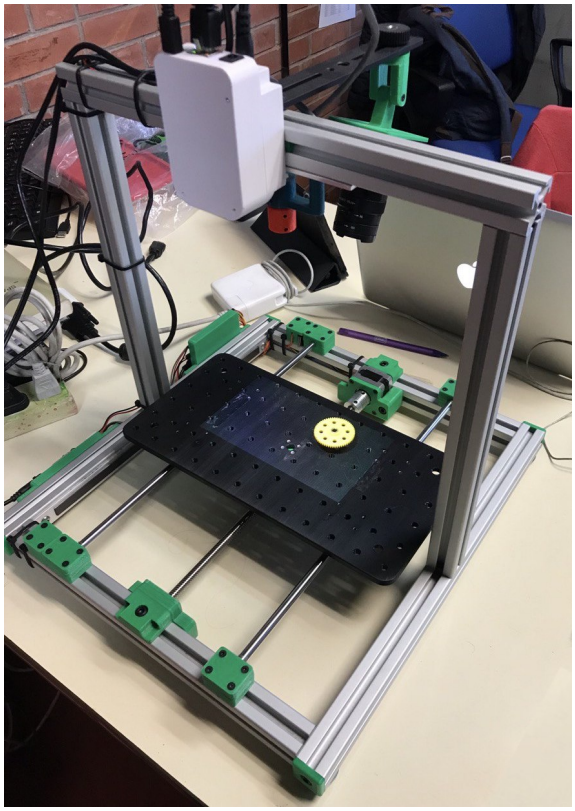


Fig. 3.22: Escáner 3D V1.

Por último se agregó una cámara adicional acoplada al nuevo arco. La cámara utilizada es una cámara USB Logitech c920 que ofrece una resolución de captura de $2304x1536$ píxeles. A esta cámara se le remplazó la carcasa para permitir su acople a la estructura y a una lente de ajuste manual como la utilizada en la RPiCam. La RPi junto con la RPiCam se reubicaron en el arco original y el proyector se colocó entre medio de ambos arcos, siempre orientado a la plataforma.

La ubicación tanto de las cámaras como el proyector se puede alterar fácilmente a discreción del usuario para adaptarse al tamaño y/o características del objeto a escanear, aunque requeriría realizar una nueva calibración del escáner.

3.5. Implementación del sistema de escaneo 3D

En esta sección se verá en detalle el hardware y software que componen el sistema de escaneo 3D (Figura 3.24).

Se verá qué función cumple cada componente y cómo se comunican. Por último se describirán los diferentes módulos que integran el software del escáner.

3.5.1. Hardware

El hardware que permite el funcionamiento del escáner es el siguiente:

1. Raspberry PI 3B (conectada a la RaspiCam y la Logitech c920)

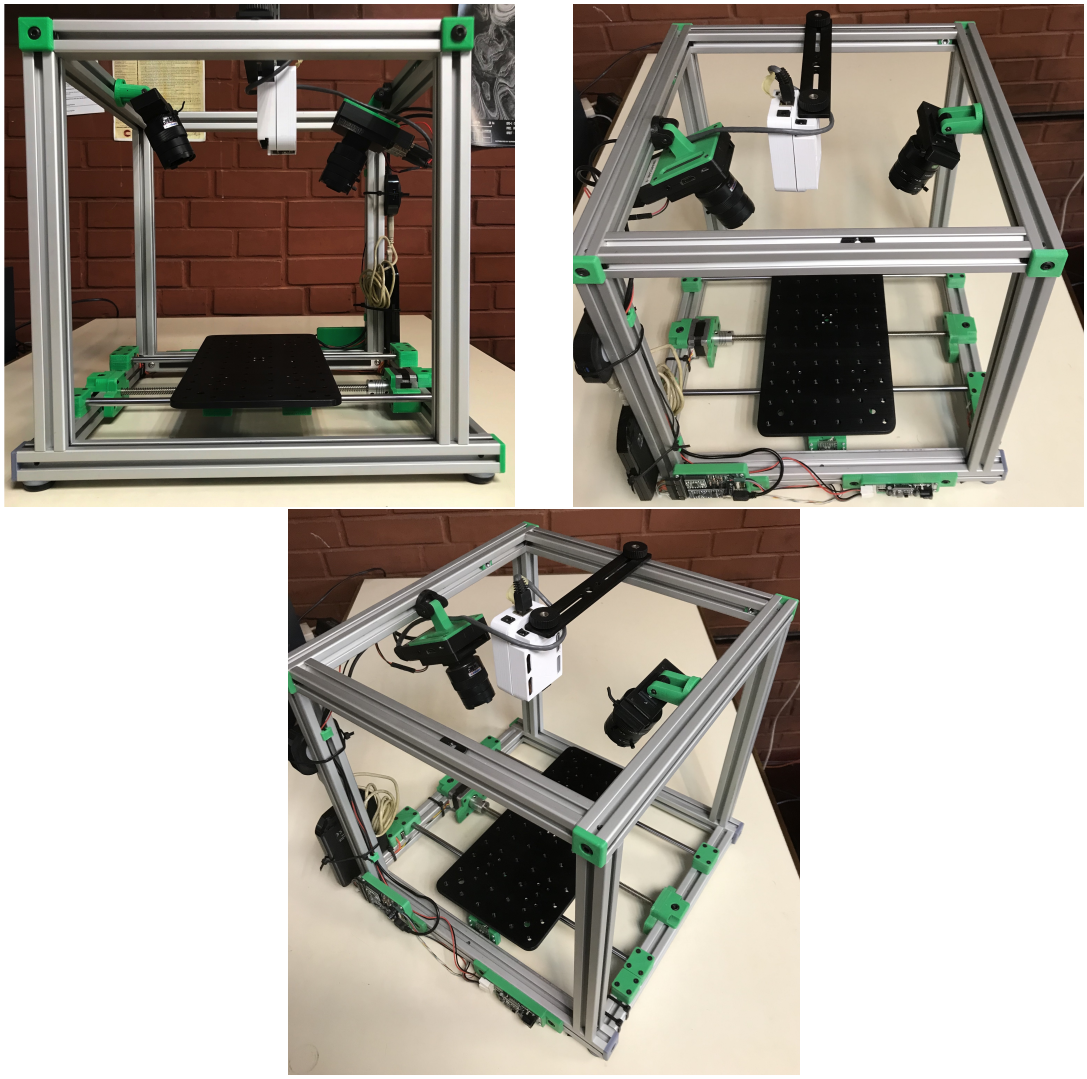


Fig. 3.23: Escáner 3D V2.

2. BeagleBoneBlack (en módulo con el pico-proyector)

3. Arduino Nano (conectada al sensor de encoder magnético y al driver de motor paso a paso)

En esta sección veremos exactamente qué función cumple cada dispositivo. También veremos cómo se preparó cada dispositivo para cumplir su función. Como mencionamos previamente la RPi va a ser la encargada de sincronizar los diferentes dispositivos, de realizar las capturas de imágenes y todo el procesamiento necesario para recuperar los puntos 3D. Por lo tanto veremos también cómo se logró la conectividad entre la BeagleBone Black y el Arduino Nano con la Raspberry Pi.

3.5.1.1. BeagleBone Black

La computadora BeagleBone Black (BBB) es la encargada de renderizar patrones de luz estructurada con codificación Gray a pedido de la RPi.

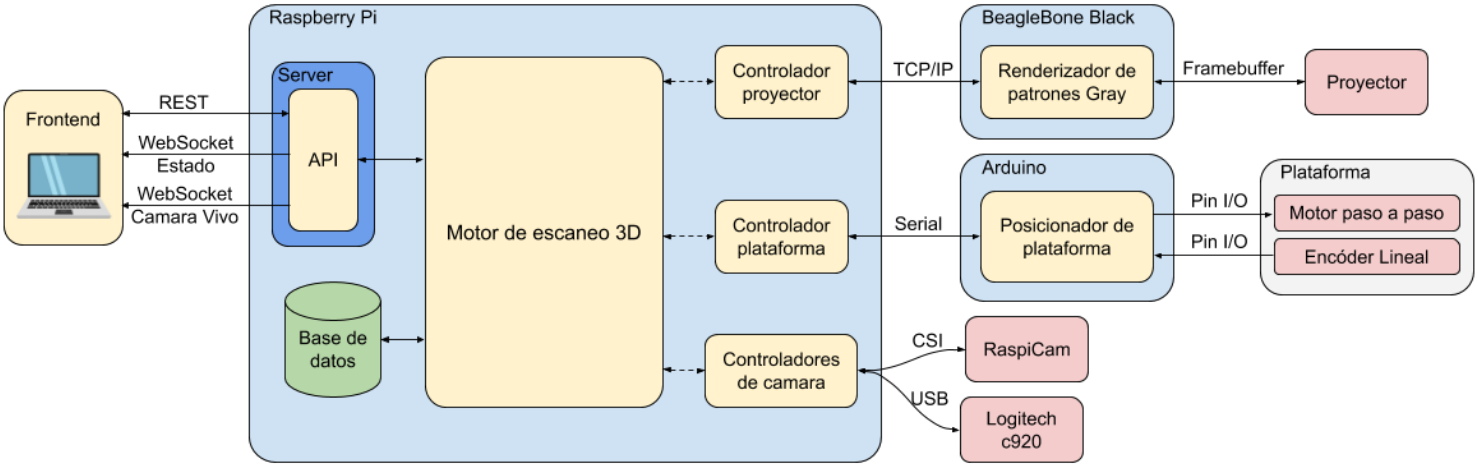


Fig. 3.24: Diagrama sistema de escaneo 3D.

Para esto se diseñó un renderizador que recibe pedidos de proyección por TCP/IP por lo que fue necesario conectar la BBB a la RPi a través de un cable Ethernet UTP-8.

3.5.1.2. Arduino Nano

La función de la placa Arduino Nano es controlar el motor paso a paso de la plataforma móvil y a su vez reportar la posición de dicha plataforma utilizando el valor registrado por el sensor del encoder lineal.

La placa Arduino Nano es una placa open-source con un microcontrolador integrado. Esta placa está equipada con un puerto micro-USB y puertos de entrada/salida (I/O) digitales y analógicos programables utilizando un lenguaje de programación basado en C.

En el diseño de Gabriel Taubin la placa Arduino Nano se encuentra conectada por medio de puertos I/O digitales a un controlador de motor paso a paso. Dicho controlador se encuentra conectado al motor paso a paso que mueve la plataforma. A su vez el sensor del encoder lineal también se encuentra conectado por puertos I/O digitales a la placa Arduino Nano. Este sensor reporta a través de dichos puertos el valor grabado en la porción del encoder sobre la que se encuentra en el momento. El valor grabado en cada posición del encoder lineal corresponde a su ubicación absoluta en milímetros dentro de la longitud del encoder. La placa Arduino Nano se conecta por micro-USB a la RPi. Esta conexión permite a ambos dispositivos comunicarse en forma serial.

3.5.1.3. Raspberry Pi 3B

La función de la Raspberry Pi 3B (RPi) es ser el controlador principal del escáner, su orquestador y procesador.

Esto significa que ejecutará el software controlador del escáner encargado de la captura de imágenes, de la comunicación con la BBB para la proyección de patrones y de la comunicación con la Arduino Nano para el control de la plataforma. A su vez va a contener el motor de escaneo 3D que contiene la lógica para el procesamiento de imágenes, la decodificación de luz estructurada, el modelado del sistema, la calibración del sistema y el procesamiento geométrico para lograr la triangulación. Por último, también va a servir la aplicación web con la que el usuario controla el escáner.

Además de conectarse con la BBB y el Arduino Nano, la RPi se conecto via Wifi a la LAN local para testeo y depuración en etapa de desarrollo, y para darle acceso al usuario a la interfaz de control en la etapa productiva.

3.5.2. Software

Para el diseño del software en general se tuvieron como objetivos importantes lograr un nivel de abstracción y modularización que permitan desacoplar el hardware del software. La idea fue desarrollar un sistema que sea simple de extender tanto a la hora de usar diferentes dispositivos como a la hora de cambiar las técnicas de triangulación y generación de correspondencias. Es por esto que se decidió dividir lógicamente el software en tres partes:

- Motor de escaneo 3D.
- Renderizador de patrones de luz estructurada.
- Posicionador de plataforma.
- Controladores.
- Interfaz de usuario.

En esta sección describiremos las implementaciones de todos los componentes de software necesarios para el funcionamiento del escáner. Para cada uno vamos a explicar su función, daremos una breve descripción de las librerías usadas, mostraremos cómo se organiza el componente y definiremos las interfaces que le permiten formar parte del sistema.

3.5.2.1. Motor de escaneo 3D

El motor de escaneo 3D se puede dividir funcionalmente en dos partes: el **motor gráfico** y el **motor de escáner**. El **motor gráfico** consiste en un conjunto de módulos encargados de:

- Modelar los diferentes modelos matemáticos utilizados en nuestro escáner: modelos de cámara estenopeica, del sistema de captura estéreo y de traslación de plataforma.
- Procesamiento de imágenes: detección de tablero de calibración, rectificado de imágenes y decodificación de luz estructurada.
- Procesamiento geométrico 3D: cálculo de rayos de cámara, intersección aproximada de rayos, estimación de pose, ajuste de rectas a conjuntos de puntos y aplicado de transformaciones afines.
- Lectura y escritura de imágenes y nubes de puntos en diferentes formatos.

El **motor de escáner** va a modelar un escáner 3D utilizando el motor gráfico. Esto significa que contiene los algoritmos y estructuras de datos que permiten:

- Aplicar los modelos matemáticos mencionados a una configuración específica de proyector, cámaras y plataforma. Es decir, efectuar la calibración intrínseca y extrínseca de cámaras y proyector, y calibración de plataforma. A su vez, lograr la persistencia de estas calibraciones.

- Organizar y almacenar los distintos tipos de capturas de imágenes realizadas por el escáner: capturas de patrones de luz estructurada, capturas de tablero de calibración y capturas calibración de plataforma.

- Interpretar y procesar las diferentes capturas para calibrar el sistema y recuperar nubes de puntos 3D según corresponda.

Para comunicarse con las cámaras, el proyector y la plataforma se implementaron interfaces genéricas simples para abstraerse de los componentes de hardware específicos. La interfaz de una cámara contempla solo la configuración de la resolución y la captura de una imagen. La interfaz de un proyector tiene que reportar al resolución del proyector y permitir el envió de pedidos de proyección de patrones Gray. Por último la interfaz de una plataforma permite la solicitud de posicionamiento de la plataforma tanto relativa como absoluta, y a su vez reporta la posición de la plataforma en todo momento.

Todo lo que implique un pedido (de captura, posicionamiento o proyección) es secuencial. Esto significa que el pedido finaliza si y sólo si el pedido se ejecutó satisfactoriamente o reportando el error correspondiente en caso de falla.

Este patrón de diseño permite la implementación de controladores que, mientras cumplan con dichas interfaces, puedan ser específicos a cualquier dispositivo y definir sus métodos de comunicación a elección.

El motor de escaneo 3D está íntegramente desarrollado en *Python* y depende de las siguientes librerías externas:

- *NumPy*: permite representar y operar con matrices multidimensionales de manera eficiente. Soporta operaciones algebraicas, matemáticas y lógicas sobre dichas matrices. Al estar escrito en *C/C++* e implementar *BLAS*, realiza dichas operaciones en tiempos no alcanzables en Python nativo.

- *OpenCV*: orientada a Visión por Computadora que implementa el estado del arte de modelado y calibración de cámaras, procesamiento de imágenes, estimación de pose, entre otros. También escrita en *C/C++*, con el objetivo principal de ser usada en tiempo real, por lo tanto es extremadamente eficiente.

- *Pillow*: permite la manipulación, visualización, lectura y escritura de imágenes. Escrita en *Python* y *C*.

- *PlyFile*: Permite la manipulación, entrada y escritura de archivos *Ply* (*Polygon File Format*).

3.5.2.2. Renderizador de patrones de luz estructurada

Éste componente se encarga de generar los patrones de luz estructurada con codificación Gray a pedido. El mismo es ejecutado por la BeagleBone Black.

El renderizador utiliza el *framebuffer* correspondiente al pico-proyector de Texas Instrument. El *framebuffer* funciona igual que una salida de vídeo normal, en el sentido que es un sector de memoria del sistema que representa los píxeles de la salida de video ordenados secuencialmente.

A su vez se desarrolló un protocolo simple que formaliza los pedidos de renderización de dichos patrones. Un pedido está integrado de el número de bit de patrón a proyectar junto con los siguientes *flags*:

- Vertical: si es *True* se codifica la fila de cada píxel, si es *False* la columna.
- Negativo: si es *True* se muestra el resultado negado (esto es requerido por el algoritmo de clasificación robusta de píxeles).

Para recibir los pedidos el renderizador actúa de servidor TCP/IP, al que se conecta un cliente que necesita enviar pedidos. El renderizador fue diseñado en *C* y *Python* y no requiere librerías externas.

3.5.2.3. Posicionador de plataforma

Se reutilizó el software posicionador de plataforma implementado[4] por Gabriel Taubin y Peter Walecki en el escáner 3D *laser slit*.

El mismo recibe pedidos por canal serial a través del puerto USB del Arduino Nano. Los pedidos soportados son el posicionamiento global y relativo de la plataforma, y a su vez el reporte de la posición actual de la plataforma en milímetros.

3.5.2.4. Controladores

Se implementaron cuatro controladores para este escáner, uno para cada uno de los siguientes dispositivos:

- Cámara Logitech c920.
- Cámara RaspiCam.
- Plataforma.
- Proyector TI.

El controlador de la **cámara Logitech c920** utiliza la librería *Open CV* que ya incorpora la captura de imagen para cámaras web y su configuración de resolución utilizando el puerto USB.

El controlador de la **cámara RaspiCam** utiliza la librería *PyCamera* diseñada específicamente para el control de RaspiCams a través del puerto *CSI* de la Raspberry Pi. A través de esa librería se puede configurar la cámara y realizar capturas de imagen con alta velocidad.

El controlador de la **plataforma móvil** utiliza la librería *PySerial* para realizar comunicación en serie a través del puerto USB donde esta conectado el Arduino Nano. El controlador inicializa la plataforma para encontrar la posición inicial de la misma y permite enviar comandos de posicionamiento a la misma siempre almacenando la posición al finalizar cada comando.

Por último el controlador del **proyector TI** se comunica a través de un *socket TCP/IP* con el renderizador presente en la BeagleBone Black. Este puede enviar pedidos de proyección de patrones específicos utilizando el protocolo específicamente diseñado que mencionamos antes, y bloquea la ejecución hasta recibir la respuesta del renderizador en caso de éxito como el error en caso de falla.

Todos los controladores se implementaron en *Python*.

[4] `http://mesh.brown.edu/3DP-2018/hw4/hw4.html`

3.5.2.5. Interfaz de usuario

La interfaz de usuario consiste de una aplicación web, y el backend (*API*) es servido por un servidor en la Raspberry Pi y se encuentra desarrollado en Python, dependiendo de las librerías *Flask* y *SocketIO*. Mientras que el frontend es servido por el mismo servidor, y renderizado por cualquier navegador web moderno. Éste fue desarrollado en *ReactJS* y requiere de soporte *WebGL* de parte del navegador web que se utilice para renderizarlo. *WebGL* permite renderizar una vista en 3D de las nubes de puntos capturadas.

El backend recibe comandos por *REST* y envía actualizaciones de estado y vista en vivo de la/s cámara/s utilizando dos *WebSockets* respectivamente. Esto permite una conexión estable y robusta entre

La aplicación web le permite al usuario calibrar el sistema de forma sencilla y configurarlo. A su vez le permite capturar un objeto 3D y visualizar la nube de puntos recuperada.

4. RESULTADOS

En este capítulo pondremos a prueba el escáner desarrollado. Primero evaluaremos su desempeño tanto en su exactitud como en su precisión, comparándolo mano a mano con un escáner comercial.

Para evaluar la exactitud y precisión del escáner vamos a realizar por un lado un estudio cuantitativo y por el otro un estudio cualitativo de su desempeño. El estudio cuantitativo se basa en escanear objetos que tengan superficies conocidas y comparar el modelo del objeto con los resultados obtenidos. Para compararlos se utiliza la distancia entre el modelo recuperado por el escáner y la superficie conocida del objeto. El estudio cualitativo consistirá en escanear objetos con superficies que contengan características complejas e interesantes y así evaluaremos visualmente cuán fiel es la representación obtenida.

En todos los estudios vamos a repetir los procedimientos utilizando el escáner de Creaform HandyScan 3D 300. Al incluir el escáner HandyScan 3D 300 en nuestros estudios podemos comparar los resultados del escáner desarrollado en esta tesis con uno de alta calidad y utilizado en la industria. A su vez podremos comparar los tiempos de captura y procesamiento que logra cada equipo.

Por último vamos a analizar de la misma forma el desempeño de la técnica de súperresolución por *space carving* tanto cuantitativa como cualitativamente.

4.1. Análisis cuantitativo

Para analizar cuantitativamente el desempeño del escáner desarrollado vamos a utilizar una técnica que suele ser utilizada en la práctica para cuantificar el error de la captura de un escáner 3D. Esta técnica consiste en escanear un objeto con una superficie conocida y modelada por una función matemática, tal que se pueda medir la distancia entre los puntos 3D capturados por el escáner y la superficie real del objeto. Al igual que en [MT12], en nuestro trabajo utilizaremos una superficie plana para realizar esta medición, específicamente una **placa de acero de referencia**, la fue mecanizada por Sinapsis Tech y Moltech S.A. con una tolerancia de error de 0,02 mm.

La superficie de un plano A es conocida y está dada por la Ecuación 4.1. Se forma utilizando un punto p_0 (Ecuación 4.2) -que pertenece al plano- y un vector n perpendicular al plano (Ecuación 4.3).

$$A = \{p = (x, y, z) \mid 0 = ax + by + cz + d\} \tag{4.1}$$

$$d = -(ax_0 + by_0 + cz_0)$$

$$p_0 = (x_0, y_0, z_0) \tag{4.2}$$

$$n = (a, b, c) \tag{4.3}$$

Si bien conocemos la función de un plano, al colocar la placa de acero de referencia sobre la plataforma del escáner, no conocemos los parámetros específicos que definen su superficie en el sistema de coordenadas de nuestro sistema. Por lo tanto vamos a encontrar los parámetros que mejor se ajusten a la nube de puntos obtenida.

Como la función del plano (Ecuación 4.1) es un polinomio lineal donde a, b, c y d son las constantes, esto nos permite definir un modelo de regresión lineal en el cual estos parámetros se estiman utilizando la nube de puntos. Como sabemos que va a haber ruido y potencialmente *outliers* en los datos capturados (ya que la resolución es finita y la correspondencia no es perfecta) vamos a utilizar un método de regresión robusto e iterativo llamado Random Sample Consensus (RanSaC) [FB81]. Utilizando RanSaC ajustamos el modelo de regresión lineal utilizando la nube de puntos, obteniendo así los parámetros de la función del plano que más se ajusta a los datos. Luego la distancia de un punto $p_1 = (x_1, y_1, z_1)$ al plano A es:

$$\frac{|ax_1 + by_1 + cz_1 + d|}{\sqrt{a^2 + b^2 + c^2}}$$

Dicha distancia entre un punto recuperado y la superficie del objeto va a ser el error de escaneo que vamos a analizar. Al calcular dicho error para cada punto recuperado, vamos a generar una distribución de errores que nos permitirán analizar y comparar el desempeño del escáner en distintos escenarios y con distintas configuraciones del sistema. Las diferentes variables externas e internas que cambiamos entre capturas son las siguientes:

- Distancia máxima de intersección aproximada: para analizar la cantidad y calidad de los puntos recuperados al cambiar el umbral de distancia máxima aceptada entre rayos intersecados.

- Luz externa: para analizar cuán robusto es el escáner contra contaminación lumínica del entorno.

- Súper-resolución utilizando plataforma: para ver qué traslación de plataforma resulta óptima y analizar el error introducido por la técnica de súper-resolución en sí.

Analizar las distribuciones con las diferentes capturas nos permitirá encontrar los parámetros del sistema óptimos, y luego, utilizando dichos parámetros podremos evaluar cuán robusto es el escáner, y cuán preciso y exacto es. Al analizar la precisión y exactitud del escáner no solo se utilizarán los puntos generados por el mismo, sino que a su vez se generarán superficies a partir de ellos utilizando *Screened Poisson Surface Reconstruction*. En el caso de las superficies generadas, tanto el ajuste de un plano como el cálculo de distribución de distancias a dicho plano contemplará solo los vértices que componen a la superficie, ignorando los polígonos que la componen.

Al mismo tiempo, escaneando la misma placa con el escáner de Creaform, y generando la distribución de distancias de la misma forma nos va a permitir comparar el escáner desarrollado con el HandyScan 3D. Esto nos dará el desempeño relativo de nuestro escáner con un escáner industrial de características certificadas. Como mencionamos previamente el fabricante del HandyScan 3D 300 certificó el dispositivo utilizado en esta tesis en su exactitud de unos 0,059 mm. Para obtener este valor el fabricante escaneó un par esferas de referencia recuperando puntos de su superficie. Luego, ajustó una esfera de mismas dimensiones a los puntos recuperados, y calculó la distancia entre dicha esfera y los puntos generando una distribución de errores de medición. A partir de estas distancias utilizó la métrica 2σ para cuantificar la exactitud, siendo σ el desvío estándar de la distribución de distancias como se menciono en la Sección 2.4. Si bien utilizar una esfera permite tener en cuenta los errores de medición en miles de direcciones del sistema de coordenadas, es

muy costoso obtener una esfera de referencia para realizar el mismo estudio. Es por esto que en esta tesis se utilizó una superficie plana como superficie de referencia.

Para cuantificar la exactitud utilizaremos la métrica 2σ de la distribución de distancias.

4.1.1. Distancia máxima de intersección aproximada

Como describimos previamente utilizamos la intersección aproximada (Ecuación 2.20) entre rayos para recuperar puntos 3D al escanear. Si bien siempre existe la intersección aproximada para dos rayos no paralelos, en nuestra implementación permitimos especificar una distancia máxima d entre los puntos p_1 y p_2 tolerada, tal que los casos en los que se sobrepase d, se ignore el punto recuperado. Esto los hacemos ya que, naturalmente, cuánto más lejanos sean esos puntos, menos sentido tiene que esos rayos realmente se correspondan al mismo punto del mundo real.

En este experimento vamos a ver como cambia el resultado al usar distintas restricciones de distancia máxima. Naturalmente cuanto menor sea la tolerancia, menos puntos se van a recuperar, es por esto que vamos a buscar un equilibrio entre la cantidad de puntos recuperados, y cuánto error se introduce al tener mayor tolerancia.

Para esto variamos d de 0,1 mm a 1,0 mm con incrementos de 0,1 mm. Observamos que para valores de $d > 0{,}4$ mm la cantidad de puntos recuperados se estabilizó. Esto quiere decir que las distribuciones eran prácticamente iguales, por lo tanto las descartamos.

Las distribuciones obtenidas se pueden visualizar en la Figura 4.1 y su valores en la Tabla 4.1.

d	Cantidad de puntos	Mínimo	Máximo	Desvió estándar
0.1	48320	-0.335	0.364	0.045
0.2	97509	-0.400	0.581	0.044
0.3	123679	-0.401	0.586	0.042
0.4	124425	-0.401	0.586	0.042

Tab. 4.1: Estadística descriptiva de las distribuciones de distancias al plano para distintos valores de distancia máxima tolerada (d). Valores expresados en milímetros.

Para ver detalladamente la variación entre $d = 0{,}1$ mm, $d = 0{,}2$ mm y $d = 0{,}3$ mm sus histogramas combinados se pueden visualizar en la Figura 4.1. A su vez para entender cómo aumenta la cantidad de puntos recuperados a medida que relajamos la restricción de distancia máxima se puede observar la Figura 4.2

Este experimento nos da la información que necesitamos para elegir un valor de d que nos de un equilibrio entre la cantidad y la calidad de los datos obtenidos. Como se puede ver en la Tabla 4.1, para $d > 0{,}3$ mm tanto la cantidad de puntos recuperados, como el desvío estándar de las distancias al plano se estabilizan. Lo que resulta contra intuitivo en las observaciones es que cuanto más restrictivo es d (es decir, cuando disminuye) no mejora la calidad de los datos obtenidos. De hecho se puede observar que la precisión disminuye (el desvío estándar incrementa) levemente. Lo que consideramos que está sucediendo es que la correspondencia entre rayos es lo suficientemente robusta como para no generar correspondencias espurias. Es para protegerse de estas correspondencias espurias que la restricción impuesta por d es más efectiva, por lo tanto, si éstas no forman una parte significativa de las correspondencias totales, no se verá una mejora evidente en su descripción estadística.

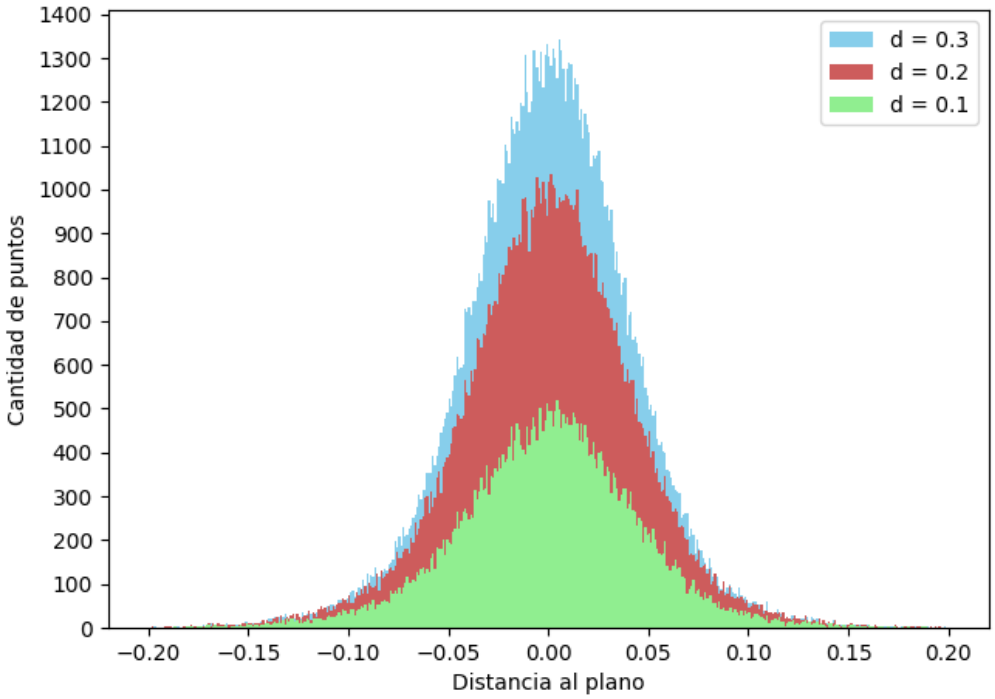


Fig. 4.1: Histograma combinado de distancias al plano para distintos valores de distancia máxima tolerada (d). Las distribuciones no se encuentran apiladas, sino que están superpuestas en orden creciente.

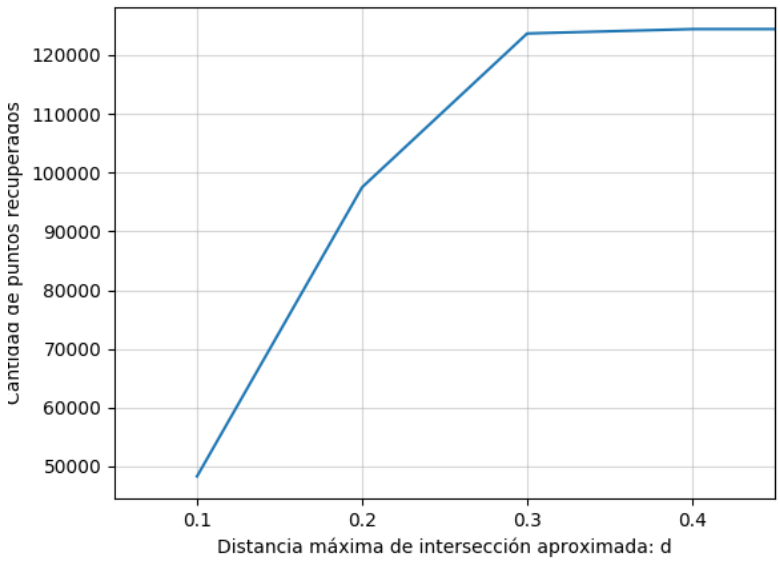


Fig. 4.2: Cantidad de puntos recuperados para distintos valores de distancia máxima tolerada (d).

Lo que sí se puede ver de forma clara es que a medida que aumentamos d aparecen outliers más distantes de la media. Estos outliers son potencialmente las correspondencias espurias de las que hablamos previamente. Es por esto que decidimos finalmente utilizar $d = 0{,}2\ mm$ en la versión productiva del escáner y para el resto de los experimentos, ya que nos va a permitir filtrar tanto correspondencias espurias como correspondencias ubicadas en zonas incorrectamente calibradas del sistema, sin disminuir drásticamente la cantidad de puntos recuperados.

4.1.2. Luz externa

En este experimento buscamos determinar cuán resistente es nuestro escáner a incidencias de luz externa. Como nuestro escáner se basa en proyectar y decodificar luz estructurada, podemos considerar como interferencia a cualquier incidencia de luz ajena al proyector del sistema. Este análisis nos permite determinar los escenarios en los cuales el escáner desarrollado es apto para realizar su función. Para el experimento en un principio escaneamos la placa de acero de referencia en la oscuridad y con luz ambiente. Las distribuciones obtenidas se pueden observar en la Tabla 4.2.

Escenario	Cantidad de puntos	Mínimo	Máximo	Desvió estándar
Sin luz	123599	-0.341	0.361	0.042
Con luz	114877	-0.337	0.300	0.041

Tab. 4.2: Estadística descriptiva de las distribuciones de distancias al plano dependiendo de la iluminación. Valores expresados en milímetros.

Si bien la cantidad de puntos recuperados fue un 10 % menor en un ambiente iluminado, la calidad de los mismos no se vio afectada. Decimos que la calidad no se vio afectada ya que la precisión obtenida es casi la misma. Esto sucede debido a que utilizamos el algoritmo de decodificación robusta de los patrones de luz estructurada, que si bien el mismo descarta más datos al decodificar, los puntos que efectivamente decodifica son de alta confianza.

Este experimento nos permite asumir que el escáner puede ser utilizado en variados escenarios de iluminación. Mientras haya iluminación ambiente pero no sea mayor a la iluminación emitida por el proyector, el escáner va a recuperar una menor cantidad de puntos pero sin reducir exactitud y precisión efectiva.

4.1.3. Súper resolución por hardware

Para estudiar cuán eficiente es la técnica de súper-resolución por hardware desarrollada vamos a realizar una serie de experimentos. Recordemos que nuestra técnica de súper-resolución consiste en mover el objeto una distancia menor al tamaño del píxel del proyector de luz estructurada.

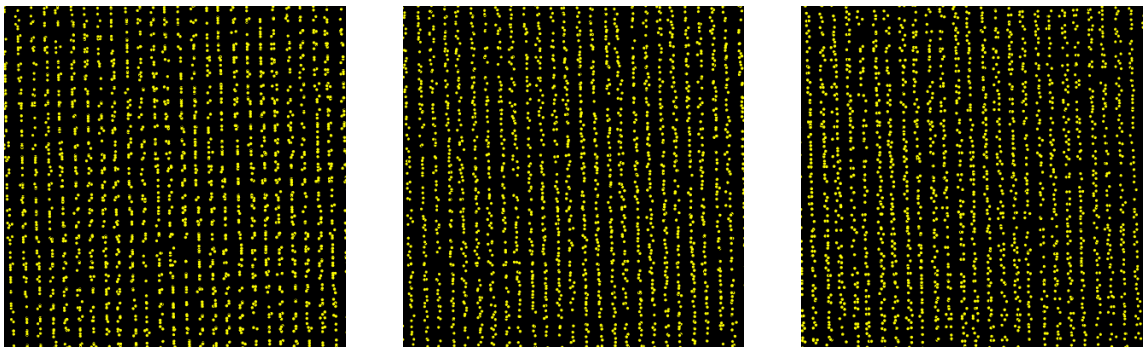
El primer experimento buscará determinar cuál es la translación entre capturas que mayor cobertura de la superficie genere. El segundo experimento analizará cuán eficiente es la técnica de súper-resolución desarrollada. Es decir, si logra aumentar la cantidad de puntos capturados sin afectar negativamente la precisión del escáner.

El tamaño del píxel proyectado va a disminuir cuanto más cerca se encuentre el objeto al proyector. Se puede estimar un tamaño de píxel promedio para la zona de trabajo usual

del escáner, que en nuestra implementación va a ser no más de $10cm$ de distancia sobre la plataforma (Figura 3.12). Como mencionamos brevemente en la Subsección 3.3.4, la distancia entre el centro de proyección del proyector y la plataforma es de aproximadamente 24 cm y el ángulo de visión del proyector es de aproximadamente 20^o. Esto significa que a la altura de la plataforma (24 cm) el píxel proyectado tiene un tamaño de aproximadamente 0,23 mm, y a la altura del fin del área de trabajo (14 cm) el píxel es de 0,13 mm. Estimamos que en promedio el tamaño del píxel proyectado dentro del espacio de trabajo es de unos 0,18 mm.

La translación que debería permitirnos cubrir de forma máxima dicha distancia utilizando n capturas es de $\frac{0,18}{n}$ mm. Como en nuestros experimentos usaremos 4 capturas para obtener súper-resolución estimamos que la translación óptima que debe realizar la plataforma entre cada captura es de 0,045 mm.

Para el primer experimento vamos a escanear múltiples veces la placa utilizando, por un lado la traslación óptima estimada, y por otro lado una translación de 0,1 mm y otra de 0,02 mm. Esto nos permitirá analizar visualmente si la translación estimada es efectivamente óptima. Los resultados se pueden visualizar en la Figura 4.3, donde se puede ver un sector ampliado del plano capturado para los diferentes valores de translación de plataforma.



(a) 0.02 mm *(b)* 0.045 mm *(c)* 0 1mm

Fig. 4.3: Nubes de puntos obtenidas para distintas traslaciones de plataforma.

Si bien esta distancia de translación logra una buena cobertura en este caso, su eficacia va a depender del tamaño del objeto escaneado. Esto quiere decir que potencialmente esta distancia de traslación puede ser ajustada dependiendo del objeto a escanear para obtener los mejores resultados posibles.

Para el segundo experimento (eficiencia de la técnica de súper-resolución) vamos a capturar la placa tanto utilizando como no utilizando súper-resolución. Calcularemos la distribución de distancias al plano para cada captura y veremos como la técnica de súper-resolución desarrollada la altera. Si al utilizar súper-resolución la distribución muestra un incremento en las distancias al plano indicaría que la técnica no logra agregar los puntos capturados para cada translación de forma correcta.

Tener en cuenta que, al estar escaneando una placa plana, y como la plataforma se mueve en forma lineal tenemos que tener cuidado de no posicionar la placa de forma tal que la traslación sea paralela al plano (Figura 4.4). Ya que en ese caso los errores causados por la técnica súper-resolución podrían no manifestarse gracias a la continuidad de la superficie del plano. Para ponerlo en otras palabras, trasladar el plano en una dirección

paralela al mismo va a resultar en el mismo plano, por lo que no estaríamos agregando nueva información al capturar con súper-resolución.

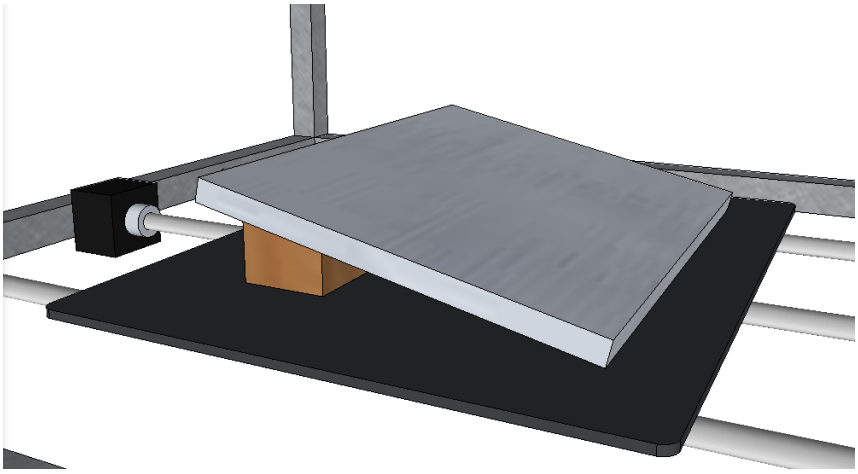


Fig. 4.4: La placa debe ser colocada de tal forma que el plano definido por la misma no sea paralelo a la dirección de movimiento de la plataforma.

Las distribuciones de distancia tanto usando como no usando súper-resolución se pueden ver en la Figura 4.5 y en Tabla 4.3.

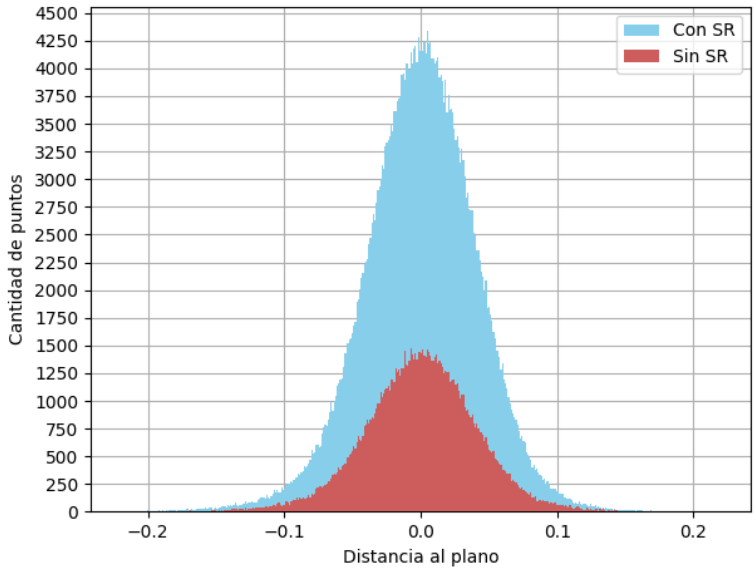


Fig. 4.5: Histograma combinado de distancias al plano para distintos valores de distancia dependiendo del uso de súper-resolución. Las distribuciones no se encuentran apiladas, sino que están superpuestas en orden creciente.

Podemos observar en la Tabla 4.3 que el desvió estándar es el mismo en ambos casos, demostrando que la técnica desarrollada no afecta negativamente la precisión obtenida.

Escenario	Cantidad de puntos	Mínimo	Máximo	Desvió estándar
Con SR	493685	-0.349	0.378	0.042
Sin SR	123599	-0.341	0.361	0.042

Tab. 4.3: Estadística descriptiva de las distribuciones de distancias al plano dependiendo del uso de súper-resolución. Valores expresados en milímetros.

4.1.4. Precisión y exactitud del escáner

En este experimento buscamos evaluar la precisión y exactitud del escáner desarrollado y a su vez de la técnica de súper-resolución por *space carving*. Al mismo tiempo compararemos los resultados obtenidos contra un escáner 3D comercial de uso profesional desarrollado por Creaform llamado HandyScan 3D 300. El escáner HandyScan 3D 300 reporta una exactitud de hasta 0,04 mm en el catálogo de escáneres de Creaform y a su vez, el HandyScan 3D 300 específico que utilizamos se encuentra certificado en su exactitud en unos 0,059 mm por el fabricante.

Para comparar de la forma más justa posible ambos escáneres vamos a capturar el plano utilizando ambos dispositivos en entornos óptimos y con parámetros de escaneo óptimos. Para el escáner desarrollado en esta tesis, un entorno óptimo es uno no iluminado de forma externa, con distancia máxima de intersección aproximada $d = 0{,}2\ mm$ y con pasos de súper-resolución de 0,045 mm. Ambos parámetros resultaron óptimos en los experimentos previos (Subsección 4.1.1 y Subsección 4.1.3 respectivamente).

En el caso del escáner Creaform HandyScan 3D 300, se configuró en su mejor resolución y al igual que antes en un ambiente oscuro de captura, ya que al igual que en nuestro sistema pero en menor medida, el escáner HandyScan 3D es susceptible a luz externa. Es importante tener en cuenta que el escáner HandyScan 3D da como resultado una malla poligonal 3D y no una nube de puntos.

Procedimos a escanear la placa plana con ambos escáneres en las condiciones óptimas mencionadas antes. Para comparar los resultados, en primer lugar acudimos nuevamente a ajustar un plano a los datos obtenidos y generando la distribución de distancias de los datos a dicho plano. En el caso de nuestro escáner como nos permite trabajar con la nube de puntos recuperada, decidimos primero generar la distribución de distancias a partir de la nube de puntos sin procesar.

A su vez generamos dos superficies a partir de la nube de puntos utilizando la técnica de reconstrucción de superficies *Screened Poisson Surface Reconstruction*. Para la primera superficie utilizamos como parámetro un valor de profundidad del octree $d = 8$ para el algoritmo de reconstrucción ya que visualmente resultó similar a la superficie obtenida por el escáner HandyScan 3D, a esta superficie la llamamos *Poisson 8*. Para la segunda superficie utilizamos un nivel de detalle $d = 9$ para aprovechar más la cantidad de puntos recuperados, a esta superficie la llamamos *Poisson 9*.

Tanto para esas superficies generadas, como para la superficie recuperada por el escáner HandyScan 3D ajustamos un plano a sus vértices y calculamos la distribución de distancias de dichos vértices al plano ajustado. Los resultados pueden verse en Figura 4.6 y Tabla 4.4.

A su vez podemos visualizar la distancia al plano de cada punto 3D recuperado para nuestro escáner en Figura 4.7 y Figura 4.8. La distancia de los vértices de la superficie recuperada por el escáner HandyScan 3D se pueden ver en Figura 4.9, de la superficie *Poisson 8* en Figura 4.10 y de la *Poisson 9* en Figura 4.11.

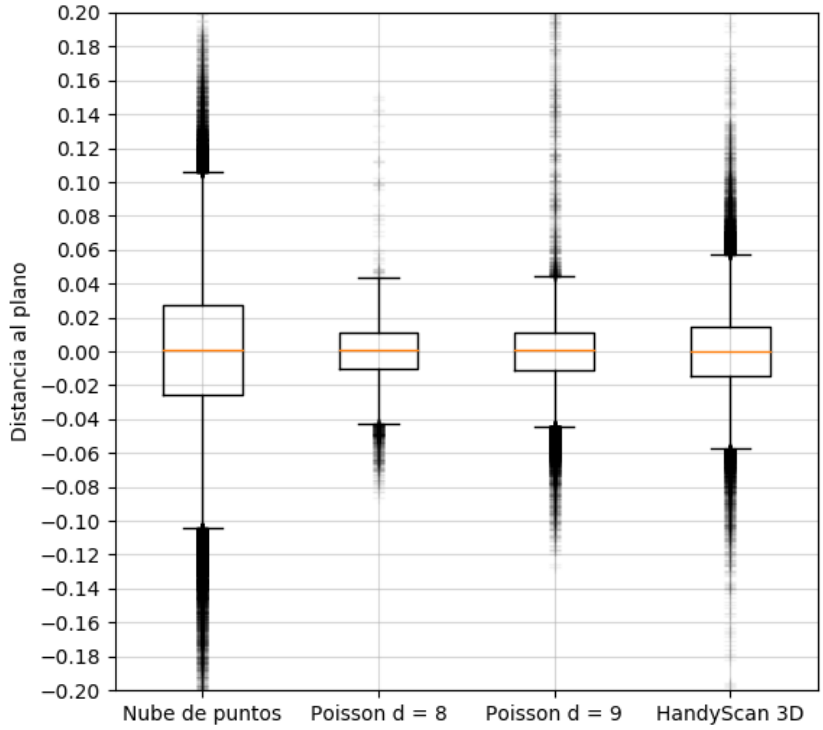


Fig. 4.6: Distribuciones de distancias al plano para ambos escáneres.

Escáner	No de puntos	Mínimo	Máximo	Desvío estándar
Nube de puntos	370264	-0.348	0.378	0.041
Poisson 8	29804	-0.086	0.15	0.017
Poisson 9	121936	-0.126	0.511	0.028
HandyScan 3D	150710	-0.353	0.354	0.024

Tab. 4.4: Estadística descriptiva de las distribuciones de distancias al plano para ambos escáneres. Valores expresados en milímetros.

Para todas las distribuciones vamos a considerar *outliers* a toda medición que se encuentre sobre $M = Q3 + 1{,}5IQR$ siendo $Q3$ el tercer cuartil e IQR el espacio intercuartil de la distribución. En los *box plot* presentados, M es representado por la línea superior a partir de la cual toda observación es considerada un *outlier*.

Como mencionamos en la Sección 2.4, la medida que se suele utilizar en la práctica para cuantificar la precisión de un conjunto de mediciones repetidas es el desvió estándar.

En la Tabla 4.4 se puede observar que en todos los casos nuestro escáner trabaja en el mismo orden de magnitud de precisión que el escáner HandyScan 3D.

En el caso de utilizar la nube de puntos recuperada por nuestro escáner sin procesar, se puede ver en Tabla 4.4 que si bien los límites son similares, el desvió estándar obtenido es casi el doble que el obtenido con el HandyScan 3D 300. Esto creemos que se da debido a

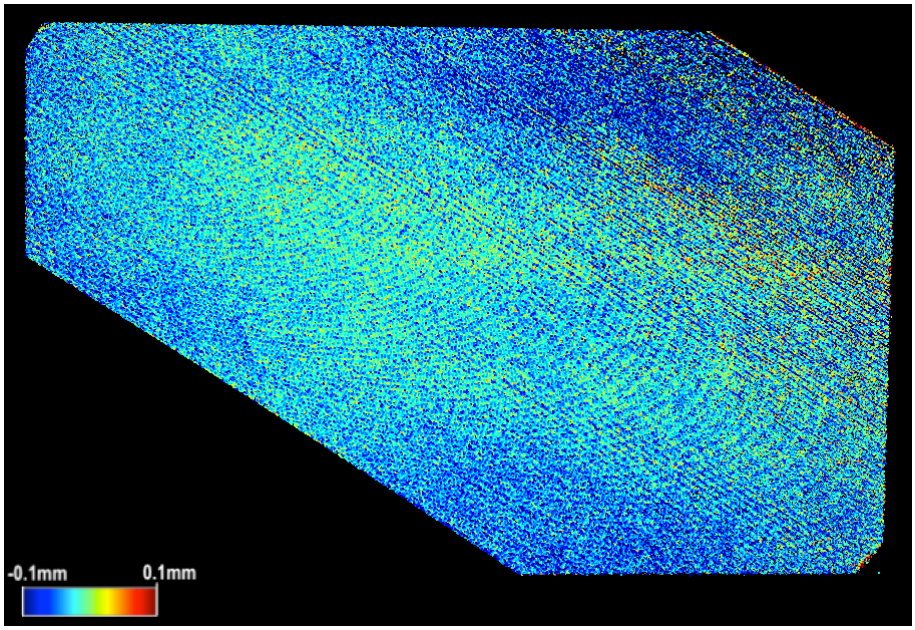


Fig. 4.7: Distancias al plano para cada punto recuperado por nuestro escáner. Vista frontal.

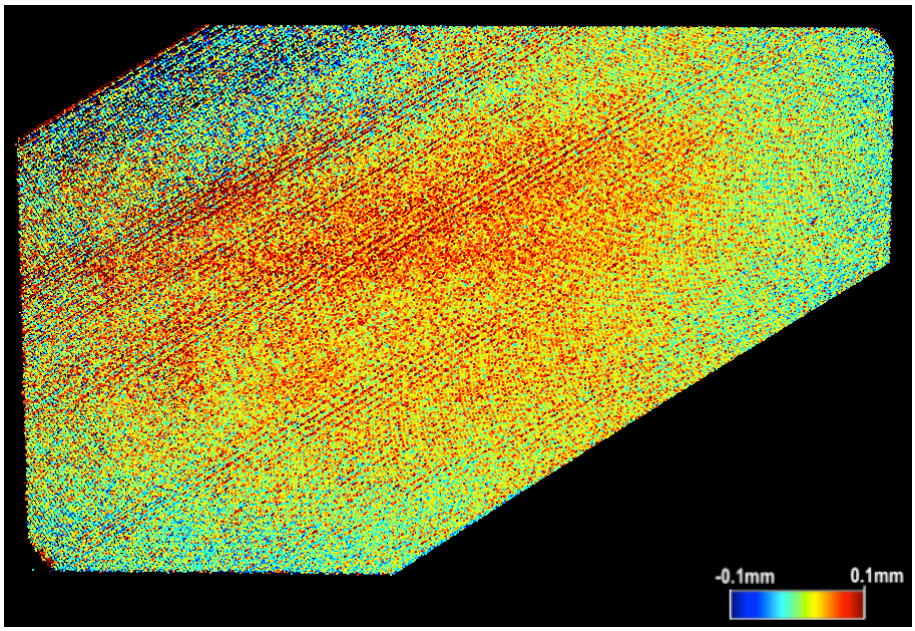


Fig. 4.8: Distancias al plano para cada punto recuperado por nuestro escáner. Vista posterior.

que el HandyScan 3D genera una superficie a partir de la nube de puntos recuperada, por lo tanto efectivamente elimina cualquier superposición de puntos en la dirección perpendicular al plano. Esto logra atenuar tanto errores de triangulación como ruido de captura, aumentando así la precisión de los resultados.

La superposición de puntos en la nube de puntos generada por nuestro escáner es evidente al ver la Figura 4.7 y Figura 4.8. Allí se puede ver como hay ciertos puntos delante del plano ajustado, y otros detrás del mismo. Es por esto que creemos utilizar

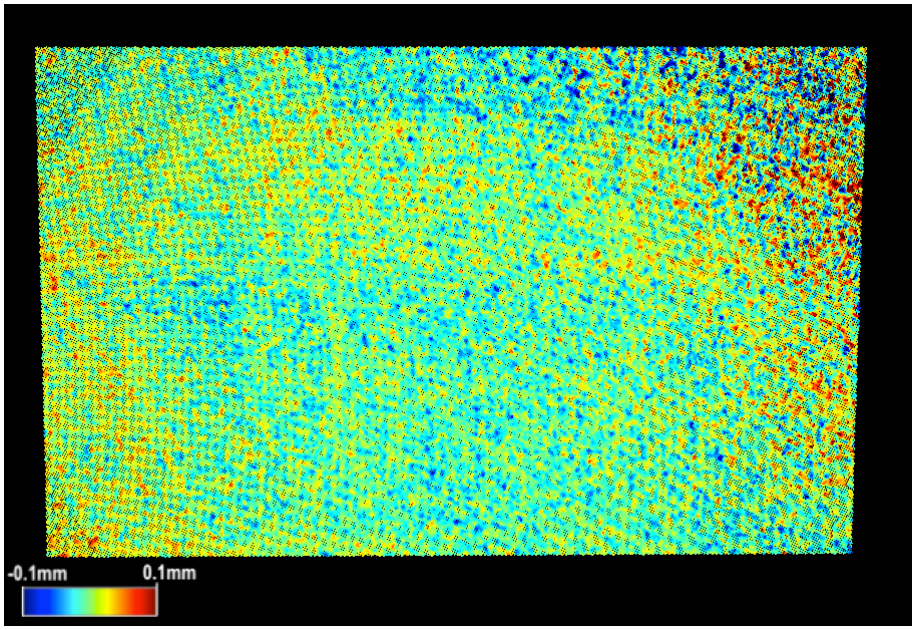


Fig. 4.9: Distancias al plano para cada vértice de la superficie recuperada por el escáner de Creaform.

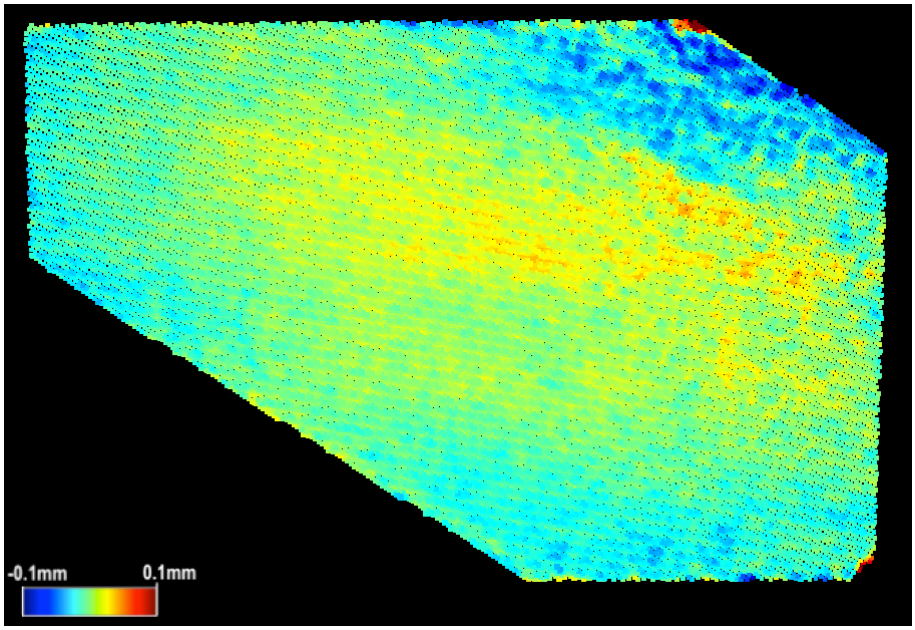


Fig. 4.10: Distancias al plano para cada vértice de la superficie generada a partir de la nube de puntos recuperada por nuestro escáner con d = 8.

una superficie generada a partir de la aproximación de la nube de puntos recuperada por nuestro escáner permite una comparación más justa entre los escáneres.

Tanto en el caso de la superficie *Poisson 8* como en la *Poisson 9*, el escáner desarrollado logró mantener la distancia máxima al plano, cercana a los 0,04 *mm* (sin considerar *outliers*). La superficie *Poisson 8* a su vez logró una precisión de 0,017 *mm*, levemente

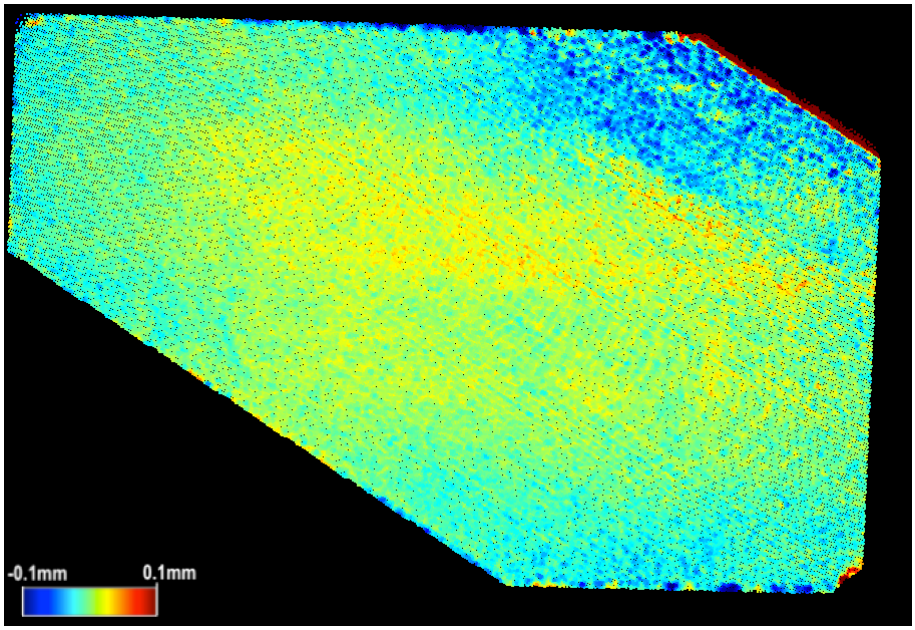


Fig. 4.11: Distancias al plano para cada vértice de la superficie generada a partir de la nube de puntos recuperada por nuestro escáner con d = 9.

mejor a la del HandyScan 3D en 0,024 mm como se puede observar en la Tabla 4.4.

Si se compara la Figura 4.9 con la Figura 4.10 y la Figura 4.11 se puede observar que en nuestro escáner las zonas de bajo desempeño son localizadas, un comportamiento poco deseado de un dispositivo de medición. Esto puede ser causado tanto por una calibración sub-óptima del escáner, como por el error de fabricación de la placa de acero.

Cuantificando la exactitud del escáner utilizando 2σ (Sección 2.4) como mencionamos previamente se puede notar que, en el caso del HandyScan 3D 300, obtuvimos una exactitud de 0,048 mm, similar a la reportada por la certificación. En el caso de nuestro escáner obtuvimos una exactitud de 0,082 mm al utilizar la nube de puntos sin procesar, 0,056 mm utilizando la superficie con parámetro de profundidad $d = 9$ y 0,034 mm si utilizamos la superficie generada con parámetro de profundidad $d = 8$. Esta exactitud obtenida a partir de nuestro escáner es competitiva, con $d = 9$ obtenemos un valor de exactitud cercano al obtenido con el escáner HandyScan 3D 300 y con $d = 8$ uno mejor.

Al mismo tiempo hay que tener en cuenta que una placa plana como la utilizada es una superficie poco desafiante de escanear por los escáneres utilizados. Es por esto que más adelante haremos estudios de superficies con mayores características para evaluar el escáner en el entorno en el que va a ser utilizado en la práctica.

4.2. Análisis cualitativo

En este análisis queremos ver cuán eficaz es el escáner desarrollado a la hora de escanear objetos con superficies complejas y detalladas. La idea es probar el escáner desarrollado en escenarios reales y desafiantes. Vamos a seleccionar objetos que nos permitan cubrir una amplia gama de texturas y formas, por ejemplo que presenten una variada gama de ángulos de curvatura, oclusiones, y detalles importantes de escala submilimétrica, entre otros.

Para llevar a cabo este análisis vamos a escanear cada objeto utilizando súper-resolución. En el caso de que un objeto necesite de varias capturas para cubrir su superficie utilizaremos ICP (Subsección 2.3.1) para unir las diferentes nubes de puntos capturadas.

Por último reconstruiremos la superficie utilizando el método de *Screened Poisson Surface Reconstruction*. Vamos a generar nuevamente dos superficies, la primera utilizando como parámetro de profundidad $d = 8$ (la llamaremos *Poisson* 8) del algoritmo de reconstrucción de superficie, y la segunda con dicho parámetro $d = 9$ (la llamaremos *Poisson* 9).

Analizaremos visualmente las superficies obtenidas, comparándolas con el objeto escaneado. Buscaremos ver cuántos y cómo se capturaron los detalles interesantes de las superficies. Luego vamos a escanear los mismos objetos con el escáner HandyScan 3D 300 y compararemos la superficie obtenida por dicho escáner con las superficies generadas a partir de la captura de nuestro escáner. Haremos un análisis visual centrándonos en los detalles mencionados antes, y a su vez calcularemos las diferencias entre las superficies obtenidas utilizando la distancia de Hausdorff.

Para calcular la distancia de Hausdorff primero alineamos tanto la superficie estimada con la superficie recuperada por el escáner de Creaform. Para esto utilizamos nuevamente el algoritmo de ICP y luego calculamos la distancia de Hausdorff entre cada polígono de la superficie Poisson 8 contra la superficie obtenida por el HandyScan 3D 300.

Los objetos elegidos son:

- Molde positivo de una dentadura humana.
- Grabado en metal con texto.
- Medalla estampada.
- Fósil del mesozoico llamado Trigonia.

4.2.1. Molde dental

El escaneo 3D es muy utilizado en el campo de la odontología. Además, las dentaduras y encías contienen muchos detalles, tipos de ángulos y oclusiones por lo que resultan ser superficies desafiantes de extraer. En este estudio se escaneó un molde positivo de una dentadura como se ve en la Figura 4.12.

Fig. 4.12: Molde de dientes.

Escanearlo completamente con nuestro escáner consistió en realizar 4 capturas: del frente, del lado izquierdo, derecho y desde debajo en la zona del paladar. Estas cuatro capturas recuperaron 4 nubes de puntos respectivamente, y estas fueron combinadas utilizando ICP. La nube de puntos resultante se utilizó para generar la superficies *Poisson* 8 y 9. Escanearlo con el escáner HandyScan 3D requirió una sola sesión que implicó desplazar manualmente el escáner para cubrir toda la superficie de los dientes y paladar de distintos ángulos de vista.

Luego, alineamos las tres superficies (la generada por el HandyScan 3D 300 y las *Poisson* 8 y 9) utilizando ICP. Una vez hecho esto calculamos la distancia de Hausdorff entre la superficie *Poisson* 8 y la superficie generada por el HandyScan 3D 300. Para visualizar los resultados capturamos todas las superficies desde ciertos ángulos que muestran las partes más interesantes de las mismas, y permiten comparar el desempeño de cada escáner.

En las Figura 4.13, Figura 4.14, Figura 4.15 y Figura 4.16 podemos ver los resultados observados desde distintos ángulos. Podemos ver que la superficie *Poisson* 8 y la generada por el HandyScan 3D 300 logran un nivel de detalle muy similar. A su vez, en ambos casos se nota que la superficie luce muy suavizada.

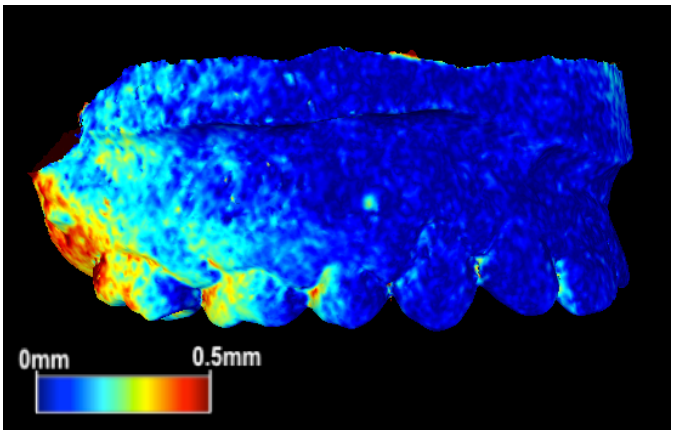
Esto se hace aún más evidente al observar la superficie *Poisson* 9 generada a partir del escáner desarrollado. En esta última el nivel de detalle es evidentemente superior, logrando recuperar desperfectos en el molde, como burbujas y excesos de yeso. A su vez los ángulos agudos presentes en la superficie del molde son recuperados de forma más fiel, esto es evidente al ver la parte dañada del molde (canino izquierdo), que dejó una rajadura en el molde. En la Figura 4.16 señalamos en rojo algunos de estos detalles. A pesar del alto nivel de detalle obtenido por *Poisson* 9, la falta de suavizado deja en evidencia el ruido en la nube de puntos. Esta se presenta como una rugosidad en la superficie observable especialmente en la Figura 4.15.

Por último observando la distancia de Hausdorff entre la superficie *Poisson 8* y la generada por el HandyScan 3D 300, se puede notar que la mayor parte las distancias se mantienen por debajo de 0,1 *mm* (azul claro) pero a su vez, hay sectores que muestran diferencias mayores a 0,2 *mm* (celeste). Esto sucede especialmente entre los dientes y en la parte posterior del molde. Estos son lugares donde nuestro escáner no logró recuperar una buena cobertura de puntos, aunque también se puede notar al observar el lateral derecho, que no se logró una alineación perfecta de las capturas.

(a) Poisson 8

(b) Poisson 9

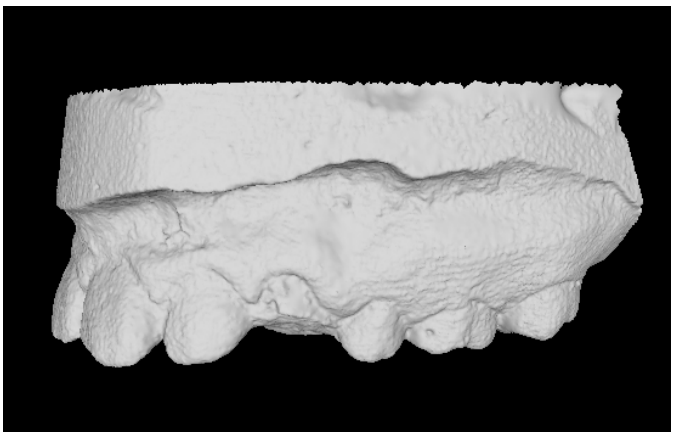
(c) HandyScan 3D 300



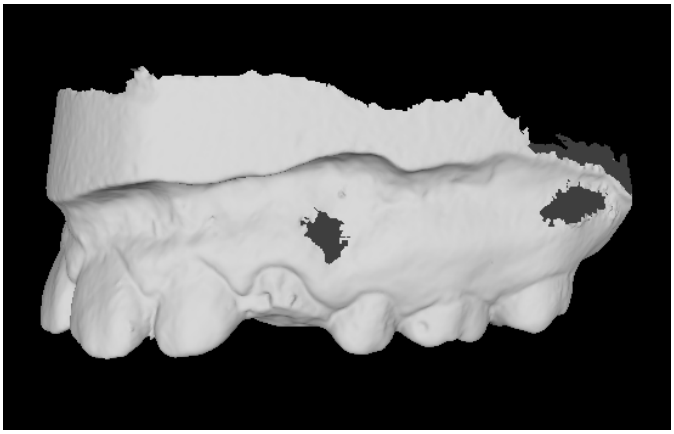
(d) Distancia Hausdorff HandyScan vs Poisson 8

Fig. 4.13: Resultados escaneo de dentadura vista lateral izquierda.

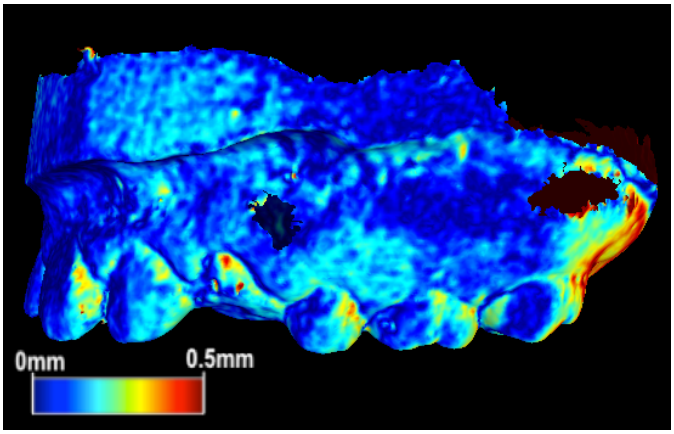
(a) Poisson 8



(b) Poisson 9



(c) HandyScan 3D 300



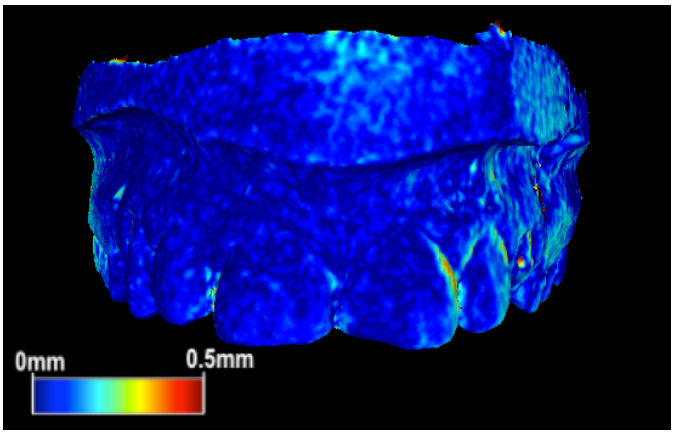
(d) Distancia Hausdorff HandyScan vs Poisson 8

Fig. 4.14: Resultados escaneo de dentadura vista lateral derecha.

(a) Poisson 8

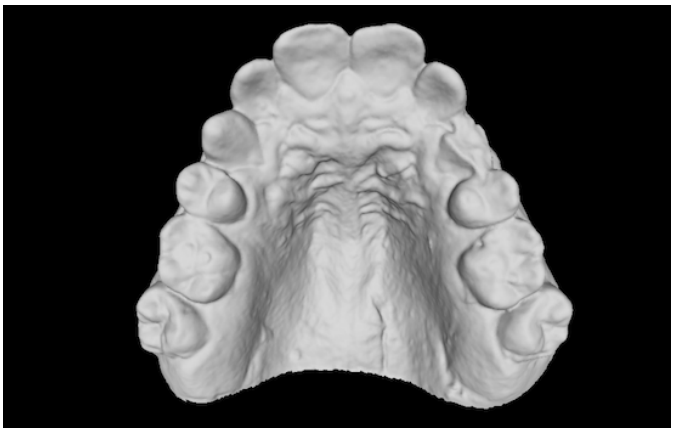
(b) Poisson 9

(c) HandyScan 3D 300

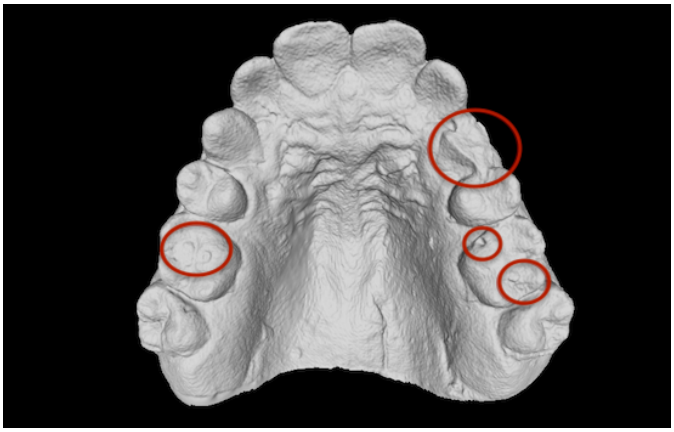


(d) Distancia Hausdorff HandyScan vs Poisson 8

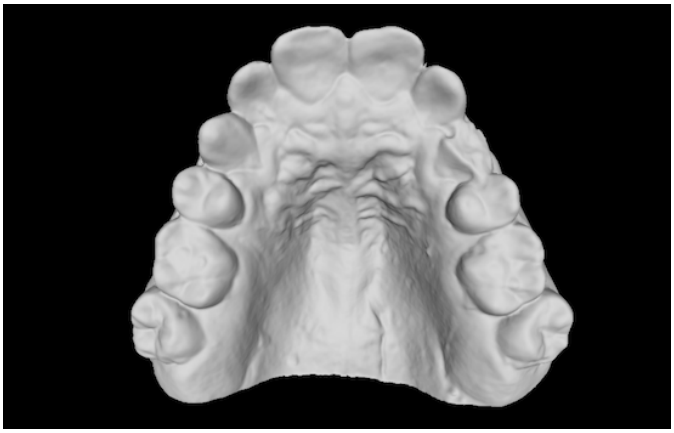
Fig. 4.15: Resultados escaneo de dentadura vista frontal.



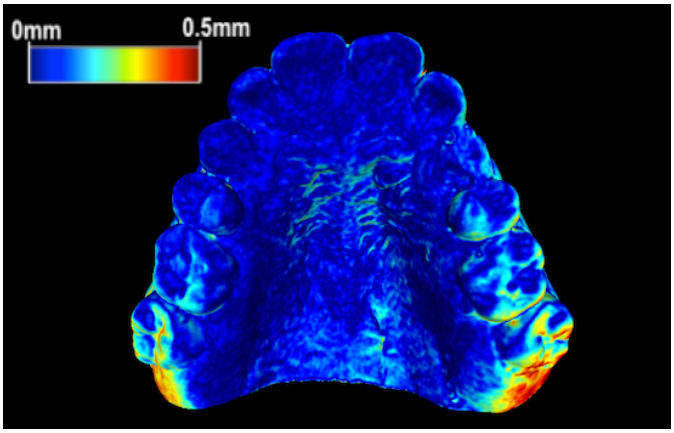
(a) Poisson 8



(b) Poisson 9. Remarcando detalles en rojo.



(c) HandyScan 3D 300



(d) Distancia Hausdorff HandyScan vs Poisson 8

Fig. 4.16: Resultados escaneo de dentadura vista inferior.

4.2.2. Grabado

En este caso el objeto de prueba es un grabado en metal de unos 8 *cm* de ancho y 2 *cm* de alto con una publicidad compuesta de letras y adornos grabados en sobrerrelieve en el metal (Figura 4.17). Tanto que la superficie sea de metal (reflectante) y que contenga detalles pequeños ofrece un desafío interesante para ambos escáneres.

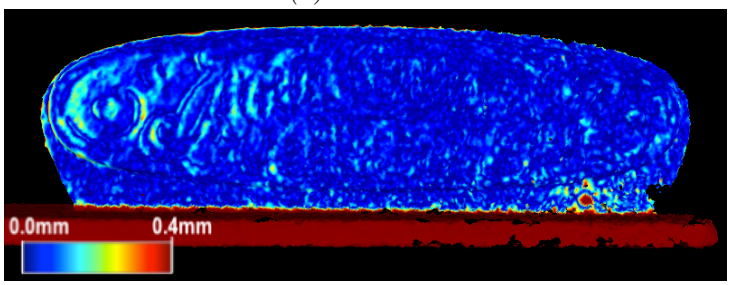
Fig. 4.17: Grabado.

En este caso nuestro escáner logró recuperar los datos necesarios con una sola captura, sin necesidad de alinear. Los resultados se pueden observar en las Figura 4.18 y Figura 4.19.

(a) Poisson 8

(b) Poisson 9

(c) HandyScan 3D 300

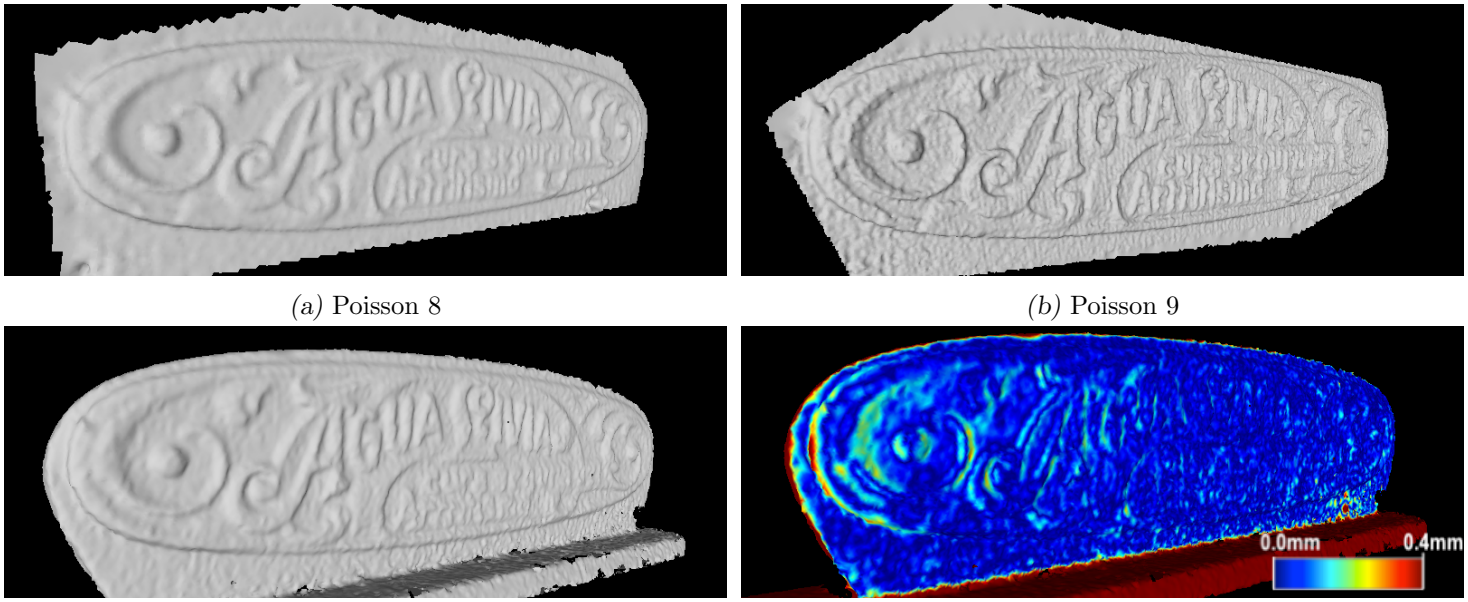


(d) Distancia Hausdorff HandyScan vs Poisson 8

Fig. 4.18: Resultados escaneo de grabado vista frontal.

Al escanear este objeto notamos que el nivel de detalle alcanzado por el escáner desarrollado en esta tesis es significativamente mayor al del HandyScan 3D 300 para esta clase de superficies. Esto se puede observar claramente al intentar leer la cita "cura segura del Artritismo", lo cual es posible únicamente en las superficies generadas a partir de nuestro escáner.

Nuevamente, *Poisson 9* muestra un nivel de detalle aún mayor que *Poisson 8*, pero es vulnerable a ruido de la nube de puntos. La diferencia de Hausdorff con *Poisson 8* en



(a) Poisson 8 *(b)* Poisson 9

(c) HandyScan 3D 300 *(d)* Distancia Hausdorff HandyScan vs Poisson 8

Fig. 4.19: Resultados escaneo de grabado vista lateral.

este caso este es menor a 0,1 mm, excepto en una pequeña área en el sector derecho del grabado. Pero incluso allí la diferencia no sube de 0,2 mm. Luego, hay un pequeño detalle en el sector inferior izquierdo que parece ser resultado de una falta de datos de captura de ambos escáneres.

4.2.3. Medalla

El modelo elegido para llevar a cabo este análisis presenta detalles más pequeños que el resto de los objetos escaneados hasta ahora (Figura 4.20).

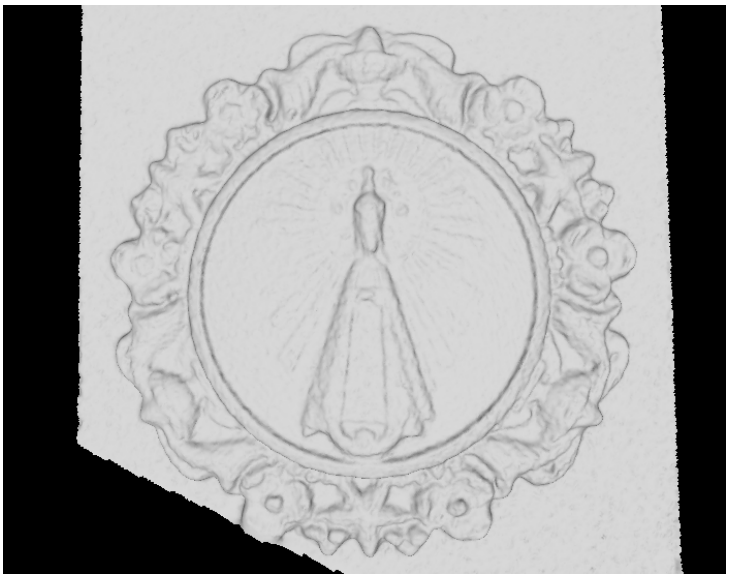
Fig. 4.20: Medalla.

El modelo es una medalla plateada con una virgen en el centro que fue fabricada con una técnica de estampado que transfiere detalles muy finos a la superficie. La misma está fabricada en metal y nuevamente decidimos escanearla sin ningún tratamiento o cubrimiento que facilite su escaneado. Al igual que con el grabado, se realizó una sola captura con nuestro escáner suficiente para cubrir el área que vamos a estudiar. Los resultados pueden verse en la Figura 4.21, la Figura 4.22 y la Figura 4.23 desde varios ángulos.

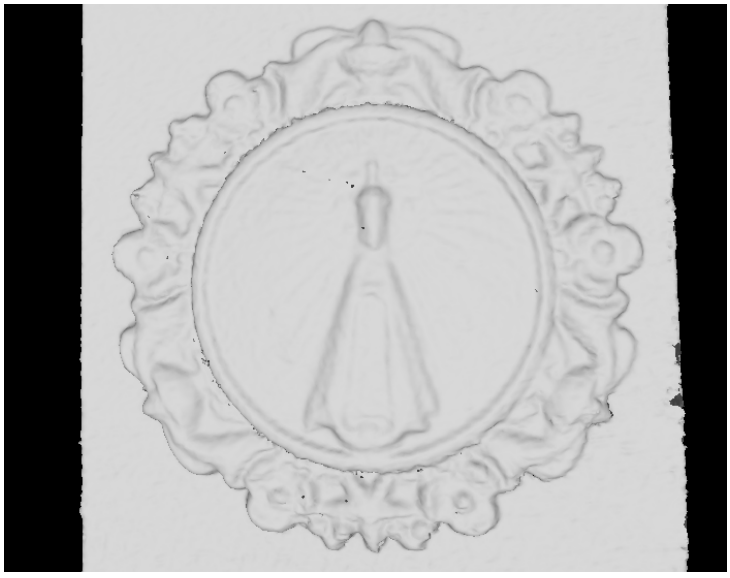
En el caso de la medalla, el HandyScan 3D 300 presentó dificultades para su captura. Si bien obtuvo una cobertura mayor, le costó capturar fielmente los bordes de la medalla. A su vez cerca del rostro de la Virgen se generaron pequeños polígonos que parecen suspendidos en el aire. Esto dejó en evidencia que el HandyScan 3D 300 no utiliza el algoritmo de reconstrucción de superficies por Poisson, ya que el mismo siempre construye una superficie continua.

A su vez notamos por primera vez un incremento importante en el ruido presente en la superficie generada por el HandyScan 3D 300, similar al ruido observado en la superficie *Poisson 9*. El nivel de detalle en este caso es menor en la superficie *Poisson 8* que en la generada por el HandyScan 3D 300, pero nuevamente la superficie *Poisson 9* alcanzó un nivel de detalle mucho mayor. Los detalles a destacar son el arreglo que rodea la medalla, el aura y las manos de la figura. Al analizar la distancia de Hausdorff en este caso las áreas que muestran diferencias notables son producto de la falta de cobertura de nuestro escáner, en especial en los bordes de la medalla, y a su vez por los artefactos generados por el HandyScan 3D 300.

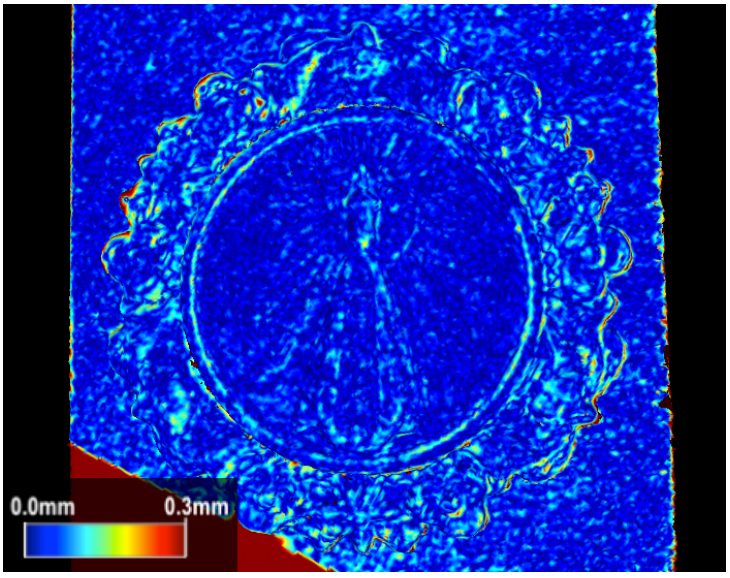
(a) Poisson 8



(b) Poisson 9

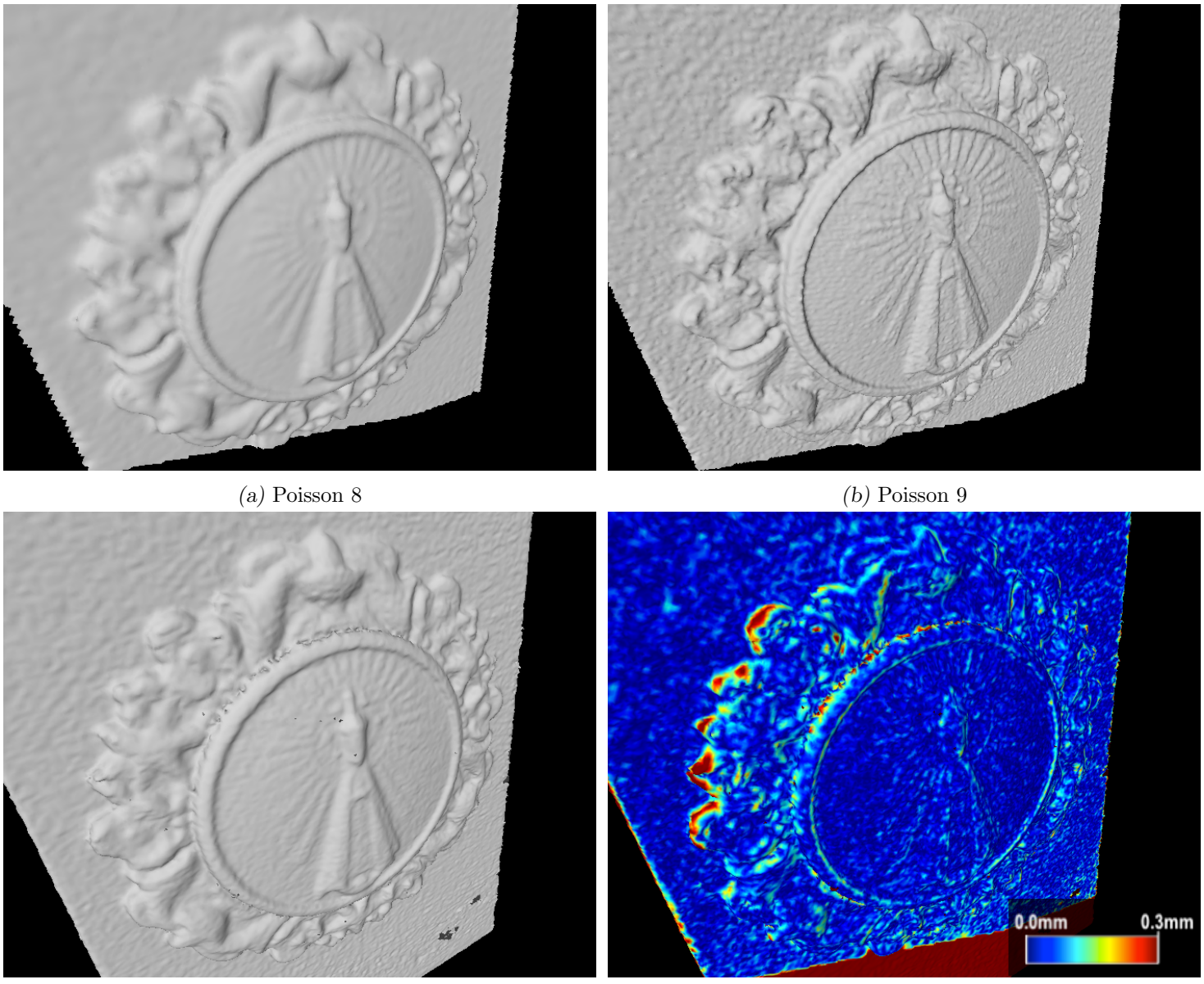


(c) HandyScan 3D 300



(d) Distancia Hausdorff HandyScan vs Poisson 8

Fig. 4.21: Resultados escaneo de medalla vista frontal.



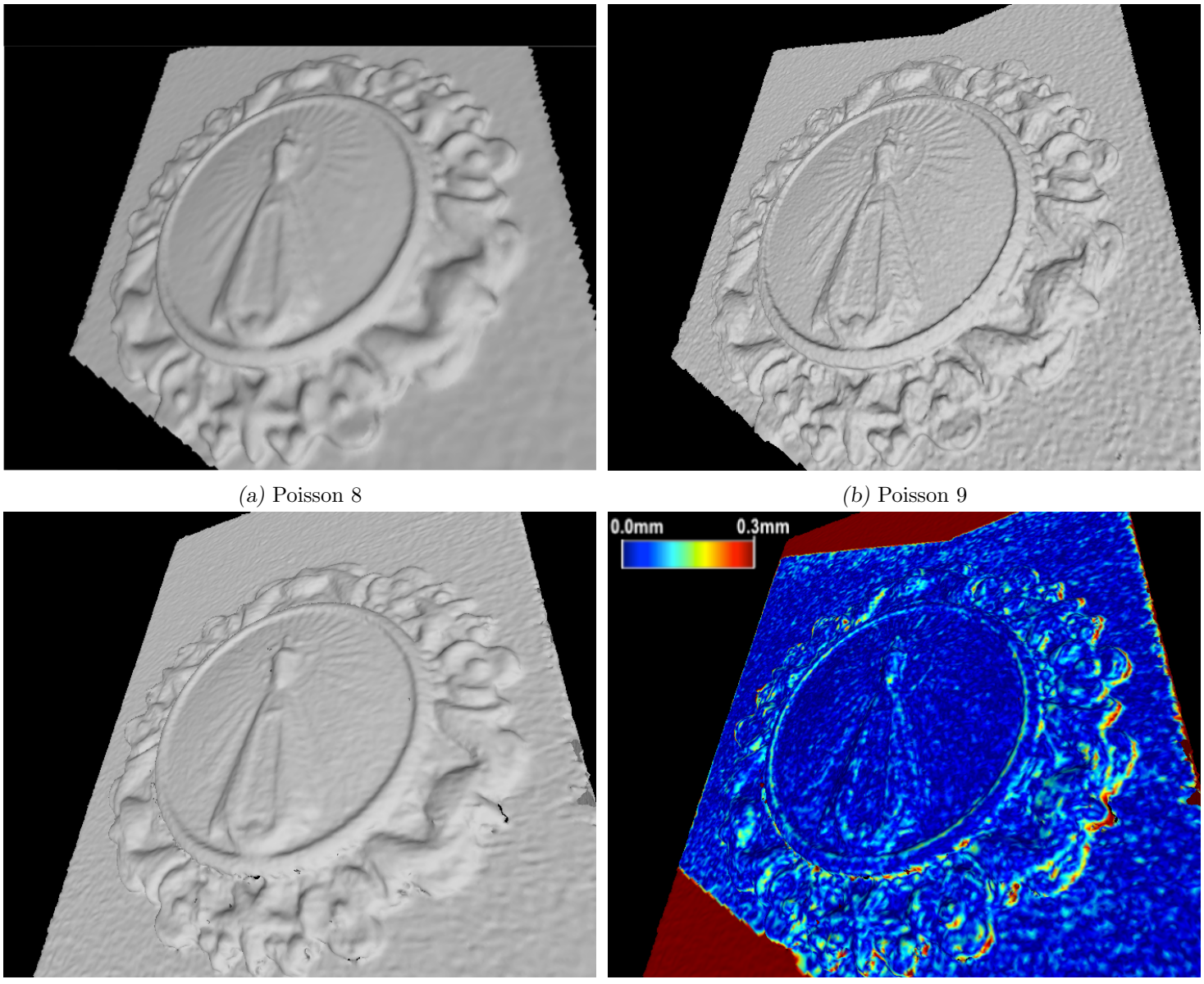
(a) Poisson 8

(b) Poisson 9

(c) HandyScan 3D 300

(d) Distancia Hausdorff HandyScan vs Poisson 8

Fig. 4.22: Resultados escaneo de medalla vista superior.



(a) Poisson 8

(b) Poisson 9

(c) HandyScan 3D 300

(d) Distancia Hausdorff HandyScan vs Poisson 8

Fig. 4.23: Resultados escaneo de medalla vista inferior.

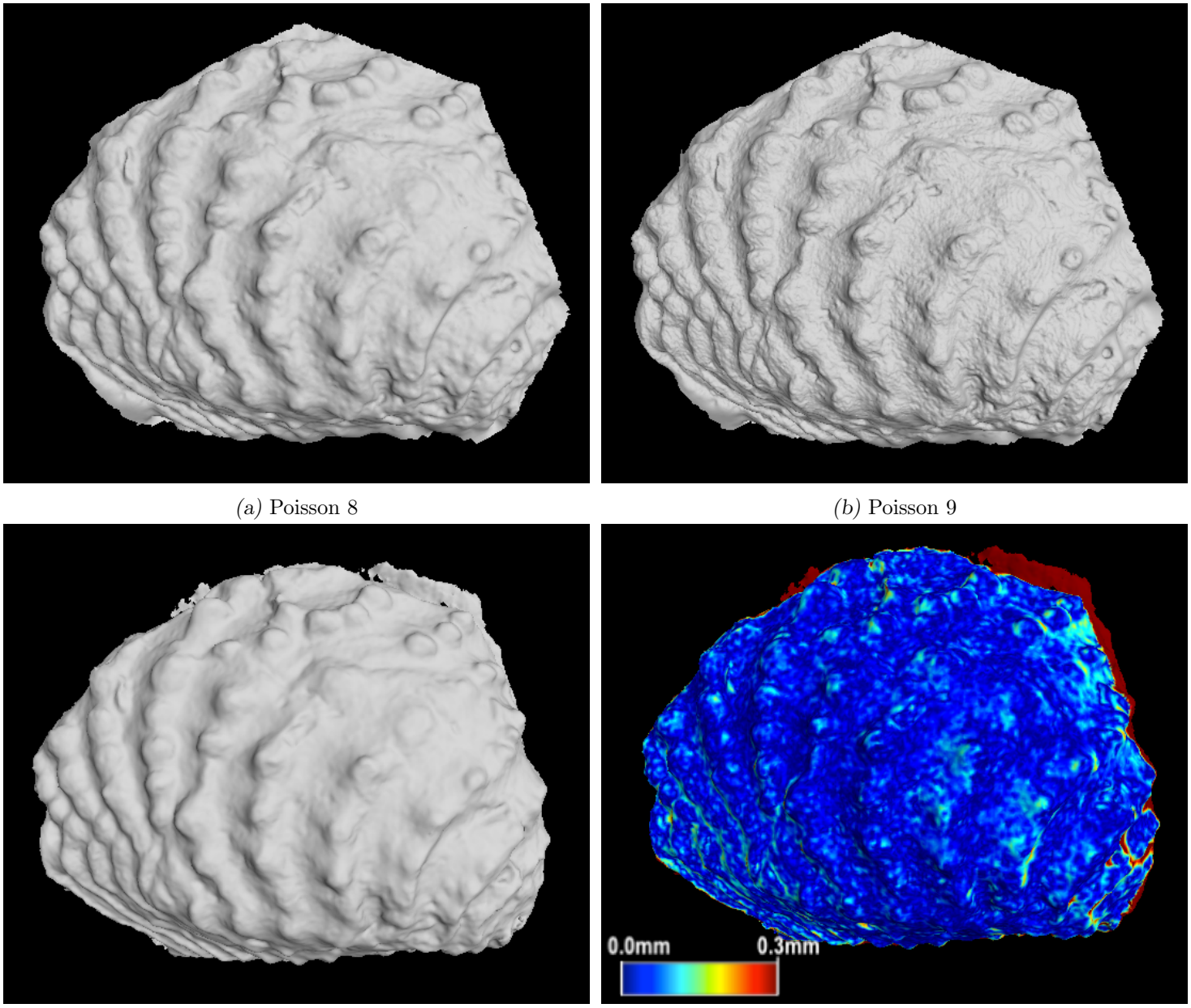
4.2.4. Trigonia

Como mencionamos en la motivación de esta tesis, unas de las áreas en las que se utiliza muy habitualmente la tecnología de escaneo 3D son la paleontología y la arqueología. Ya que, la digitalización 3D de piezas antiguas permite su manipulación y estudio sin poner en riesgo la integridad física de las mismas. Es por esto que decidimos utilizar ambos escáneres con un fósil animal antiguo llamado Trigonia que contiene una superficie rugosa, con varias imperfecciones y formas interesantes (Figura 4.24).

Fig. 4.24: Trigonia.

Realizamos una sola captura con nuestro escáner, sin necesidad de alineación, y nuevamente lo comparamos mediante la distancia de Hausdorff con la malla obtenida por el HandyScan 3D 300. Los resultados se pueden observar en la Figura 4.25.

Si bien realizar una sola captura no logró una perfecta cobertura de los laterales, los resultados son buenos. En primer lugar, el nivel de detalle que se puede recuperar en la superficie *Poisson 8* es mayor que el generado por el HandyScan 3D 300. En segundo lugar en el caso de la superficie *Poisson 9* el nivel de detalle obtenido es destacable, el escáner logró capturar orificios en la superficie causados por roturas que no se pueden detectar en la superficie generada por el HandyScan 3D 300.



(a) Poisson 8

(b) Poisson 9

(c) HandyScan 3D 300

(d) Distancia Hausdorff HandyScan vs Poisson 8

Fig. 4.25: Resultados escaneo de fósil vista superior.

4.3. Tiempos de captura

En esta sección veremos los tiempos de captura 3D logrados por nuestro escáner. Para esto vamos a dividir la captura 3D en dos partes: la etapa de adquisición y la etapa de procesamiento. La etapa de adquisición consiste en la captura de todas las imágenes requeridas por la técnica de escaneo utilizada e incluye los tiempos necesarios para sincronizar dichas capturas con la proyección de los patrones de luz utilizados. La etapa de procesamiento abarca la detección de patrones, la generación de rayos y su triangulación y la alineación de puntos 3D generados en caso necesario.

El escáner 3D desarrollado en su configuración de mayor resolución logra completar la etapa de adquisición en promedio en *47 segundos*, mientras que la etapa de procesamiento tarda en promedio *1 minuto 18 segundos*. Esto suma un total de *2 minutos 5 segundos* para recuperar una nube de puntos 3D. En el caso de aplicar la técnica de súper-resolución este tiempo se multiplicara por la cantidad de particiones utilizadas *cpart*. En el caso de usar $cpart = 4$ sumaria un total de *8 minutos 20 segundos*.

En el caso del escáner HandyScan 3D 300 al requerir un movimiento manual del mismo los tiempos pueden variar arbitrariamente. Se utilizará como medida el promedio que llevó escanear cada objeto utilizado para el análisis cualitativo. El promedio de tiempo aproximado que tomó la etapa de adquisición para el HandyScan 3D 300 es de aproximadamente *1 minuto*. Mientras que la etapa de procesamiento, que en el caso del HandyScan 3D 300 consiste en la alineación de las capturas realizadas toma aproximadamente unos *45 segundos* y depende de la cantidad de puntos de referencia utilizados. Esto suma un total de *1 minuto 45 segundos*.

Hay que tener en cuenta que en el caso del HandyScan 3D 300 la captura 3D lograda en el tiempo mencionado cubre toda la superficie objeto barrida por el usuario, mientras que en el escáner desarrollado la captura solo recupera puntos 3D desde un ángulo fijo que depende de la ubicación de las cámaras y el proyector. Esto significa que para cubrir íntegramente la superficie del objeto se necesitan potencialmente mas capturas, por ejemplo, la medalla requirió una sola captura con súper-resolución, mientras que los dientes cuatro.

4.4. Análisis de escaneo volumétrico por *space carving*

Para analizar el desempeño de la técnica de escaneo volumétrico por *space carving* vamos a utilizar los mismos métodos que utilizamos para el análisis del desempeño del escáner 3D.

Por el lado del análisis cuantitativo vamos a escanear la placa de acero y estudiar cómo afecta al desempeño la variación del único parámetro de la técnica: el tamaño mínimo de bloque de tallado para encontrar el óptimo. Para poder comparar los resultados obtenidos para cada tamaño de bloque mínimo analizado vamos a generar múltiples superficies a partir del volumen tallado y las analizaremos utilizando las mismas técnicas utilizadas antes. Es decir, se utilizarán los vértices de la superficie para ajustara un plano y luego calcular la distancia al mismo.

Para el análisis cualitativo se escaneará el mismo grabado que capturamos antes utilizando esta técnica de tallado volumétrico.

4.4.1. Análisis Cuantitativo

En este experimento buscamos analizar el desempeño de la técnica de escaneo volumétrico por *space carving* utilizando la placa de acero de referencia.

Esta técnica utiliza los rayos de cámara generados en la etapa de triangulación del escáner para tallar un cubo. El único parámetro $minBoxSideSize$ de la técnica define cuan granular es el tallado, es decir, cuan profundo puede ser el octree utilizado. Es por ello que en primer lugar se buscará encontrar el parámetro $minBoxSideSize$ óptimo para el escáner diseñado para luego analizar la precisión y exactitud del resultado obtenido.

Para analizar el resultado obtenido para cada *minBoxSideSize* probado, se necesita generar a partir del tallado algún modelo 3D que pueda analizarse.

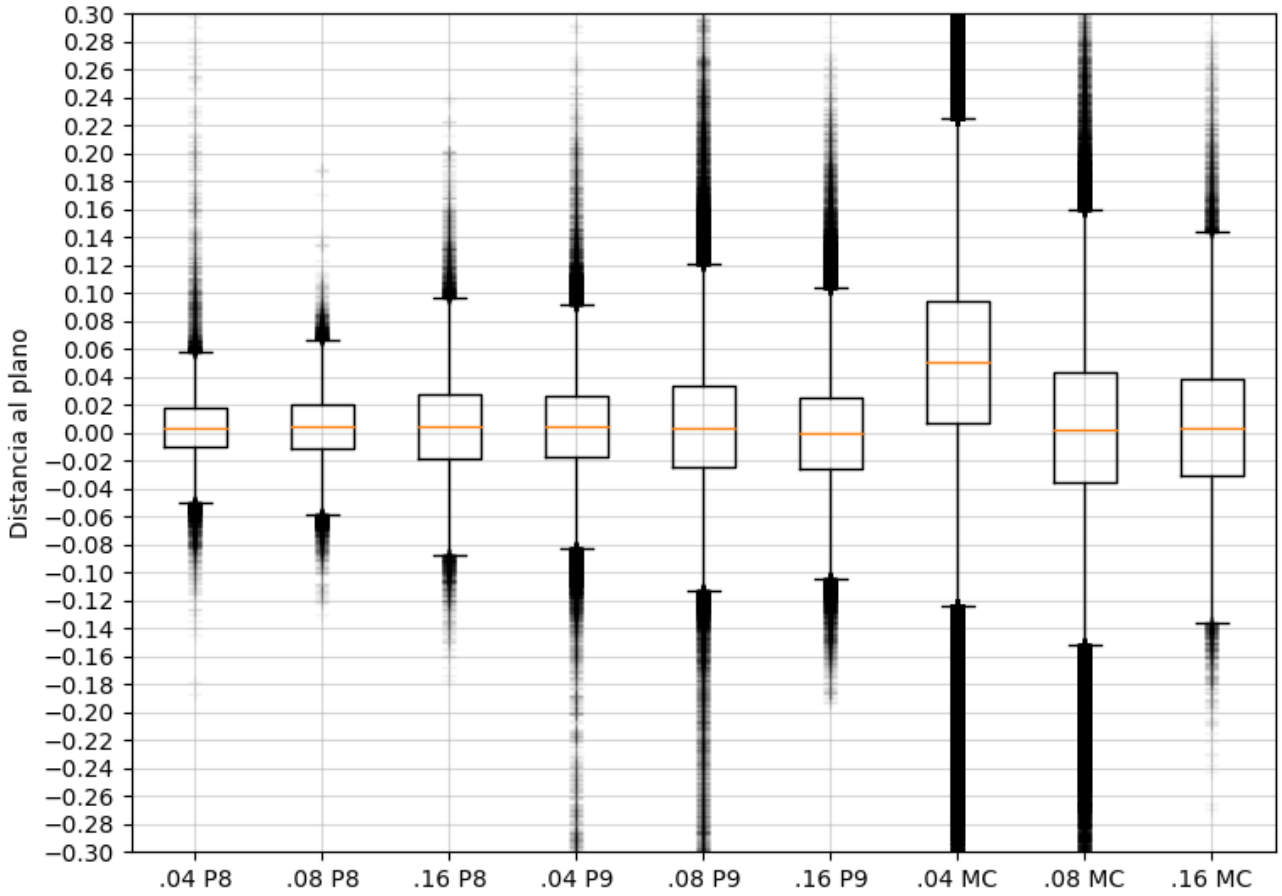


Fig. 4.26: Distribuciones de distancias al plano para cada superficie.

Como vimos en la Subsección 3.3.5 puede ser utilizada para generar superficies utilizando *Marching cubes* o *Screened Poisson Surface Reconstruction*. En el caso de este experimento, por cada *minBoxSideSize* se generará una superficie utilizando *Marching cubes* y dos utilizando *Screened Poisson Surface Reconstruction* con parámetros 8 y 9 de profundidad respectivamente. La primera será referida como MC mientras las otras dos serán llamadas $P8$ y $P9$.

Como mencionamos en el estudio de la técnica de súper-resolución por hardware, buscamos obtener datos con mayor resolución que el proyector nos provee. Es por esto que vamos a analizar valores de *minBoxSideSize* de granularidad más fina que el tamaño de píxel de proyector (tomando el tamaño de dicho píxel definido en la Subsección 4.1.3).

Con ese criterio, los valores de *minBoxSideSize* evaluados son: 0,18 mm, 0,09 mm y 0,045 mm.

Hay que tener en cuenta que al ser *minBoxSideSize* utilizado como cota mínima en el algoritmo de la técnica, el tamaño efectivo de la granularidad de tallado difiere. En este caso para cada uno de los valores de *minBoxSideSize*, el tamaño de cubo mas chico tallado fue 0,16 mm^2, 0,08 mm^2 y 0,04 mm^2 respectivamente, y es con este tamaño

con el que identificaremos a cada instancia de ahora en más utilizando *.16*, *.08* y *.04* respectivamente. Las distribuciones obtenidas para cada superficie generada se pueden observar en la Figura 4.26 y la Tabla 4.5. A su vez se puede observar el tallado 3D obtenido utilizando las distintas granularidades en Figura 4.27.

Granularidad	Superficie	Cantidad de puntos	Mínimo	Máximo	Desvió estándar
0.04 mm	P8	71945	-1.660	0.706	0.072
0.04 mm	P9	303862	-1.791	0.367	0.093
0.04 mm	MC	2367145	-1.812	0.473	0.092
0.08 mm	P8	69830	-1.728	0.188	0.077
0.08 mm	P9	317447	-1.832	0.388	0.100
0.08 mm	MC	426941	-1.865	0.388	0.095
0.16 mm	P8	70393	-1.826	0.240	0.089
0.16 mm	P9	283168	-1.848	0.338	0.061
0.16 mm	MC	88741	-1.870	0.341	0.091

Tab. 4.5: Estadística descriptiva de las distribuciones de distancias al plano para cada combinación granularidad y algoritmo de reconstrucción de superficie usado.

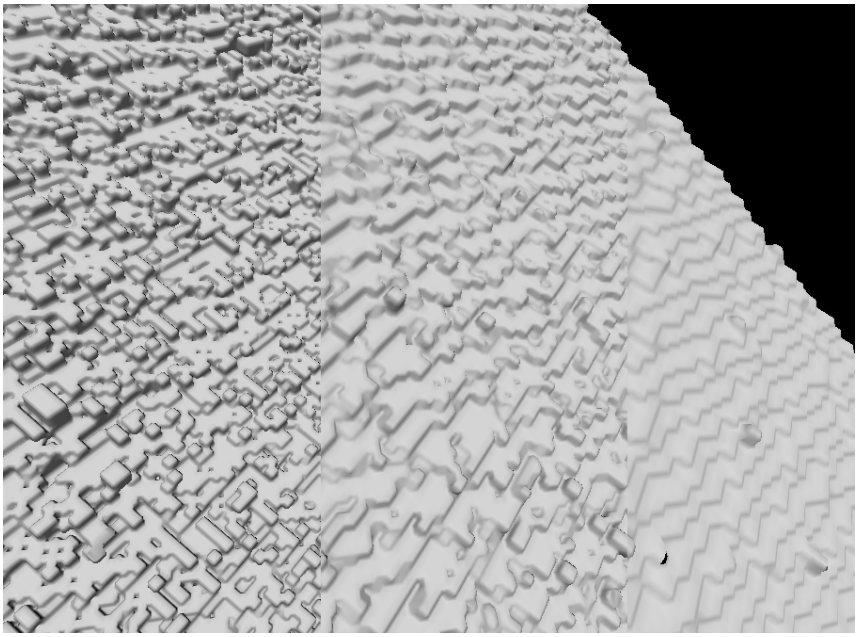


Fig. 4.27: Tallado del plano para distintas granularidades. De izquierda a derecha: 0,04 *mm*, 0,08 *mm* y 0,16 *mm*. Superficie generada con *Marching Cubes*.

Observando la Figura 4.26 dos distribuciones se destacan como óptimas: 0,04 *mm* con *Poisson* 8 y 0,08 *mm* con *Poisson* 9. Estas distribuciones logran una precisión de 0,072 *mm* y 0,077 *mm* respectivamente, y por lo tanto una exactitud de 0,144 *mm* y 0,154 *mm* con métrica 2σ.

A su vez se puede notar que las superficies generadas utilizando *Marching Cubes* se desempeñan significativamente peor que las generadas utilizando *Screened Poisson Surface Reconstruction*, y esto sucede ya que *Marching Cubes* genera polígonos dentro de los límites

de todos los cubos borde, mientras que *Screened Poisson Surface Reconstruction* no solo utiliza un punto por cubo, sino que alisa la superficie generada.

Veremos más adelante en el análisis cualitativo cuán fiel es la representación que se logra de una superficie más compleja que un plano utilizando dichos parámetros.

4.4.2. Análisis Cualitativo

Para analizar cualitativamente el desempeño de la técnica, la aplicamos al escaneo 3D del Grabado que realizamos para el análisis cualitativo del escáner 3D desarrollado.

Al escanear el grabado utilizando la técnica de escaneo volumétrico por *space carving* se decidió hacerlo con las granularidades (parámetro $minBoxSideSize$) que mostraron mejores resultados en el análisis cuantitativo de la Subsección 4.4.1. Como mencionamos en esa misma Subsección, el valor de $minBoxSideSize$ no define exactamente el tamaño del cubo mínimo tallado, sino una cota. Efectivamente los tamaños de cubo mínimo tallado para este experimento son: 0,06 mm y 0,11 mm.

A partir de ambos tallados se generaron tres superficies utilizando los siguientes métodos: *Marching Cubes*, *Screened Poisson Surface Reconstruction* con profundidad 8 y con profundidad 9. Para la visualización se amplificará el área central del grabado. Los resultados pueden observarse en la Figura 4.28.

En el caso de las superficies generadas con *Screened Poisson Surface Reconstruction* de profundidad 8, se obtuvo un resultado similar al obtenido usando Poisson con profundidad 8 con la nube de puntos pero con un levemente mejor nivel de detalle en especial en los ángulos más marcados al utilizar granularidad de 0,06 mm.

En el caso de utilizar *Marching Cubes* se nota como el algoritmo considera todos los cubos a la hora de generar la superficie sin generar ningún tipo de limpieza o suavizado, la superficie generada exhibe de forma muy explícita el tallado por lo que cuesta distinguir el grabado. En el caso de la generada con la granularidad 0,06 mm se nota que la densidad de rayos de cámara utilizados es demasiado baja comparado con la granularidad de cubos tallados, por lo que quedan cubos no tallados simplemente por estar ubicados en lugares vacíos entre los rayos. Esto genera una superficie de muy inferior calidad.

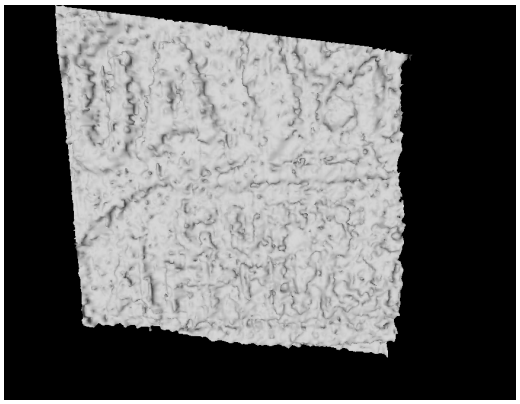
Las superficies generadas con *Screened Poisson Surface Reconstruction* de profundidad 9 logran ser un punto intermedio entre las generadas con profundidad 8 y las generadas por *Marching Cubes*, lo malo es que siguen exhibiendo los artefactos del tallado mencionados anteriormente.

4.5. Conclusiones

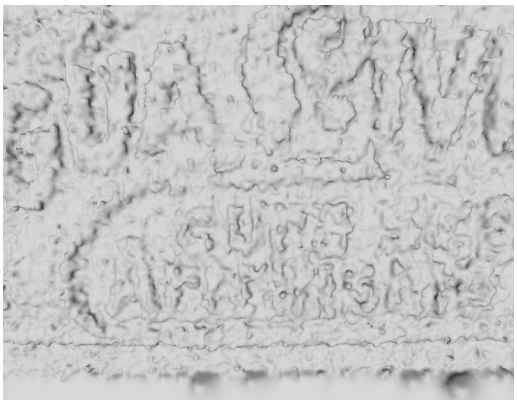
El estudio cuantitativo nos permitió medir la precisión y exactitud (en términos de la métrica 2σ) del escáner dependiendo del resultado que se busque. Si se desea como resultado una nube de puntos, la precisión estimada es de hasta 0,041 mm mientras que la exactitud es de hasta unos 0,082 mm. Si se desea de resultado una superficie en la que se obtenga un nivel de detalle alto se puede lograr utilizando la técnica *Screened Poisson Surface Reconstruction* con profundidad $d = 9$ logrando una precisión de hasta 0,028 mm y una exactitud de 0,056 mm. Mientras que si se desea disminuir el ruido en la captura manteniendo un nivel de detalle bueno, se puede usar $d = 8$, obteniendo una precisión de 0,017 mm y exactitud de 0,034 mm. Todo estos resultados reportados de forma resumida en la Tabla 4.4.

(a) 0,06 *mm* Poisson 8

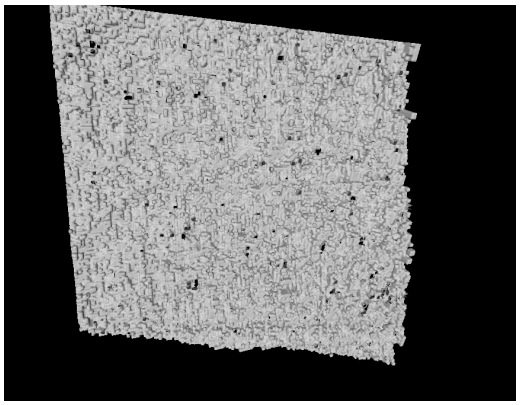
(b) 0,11 *mm* Poisson 8



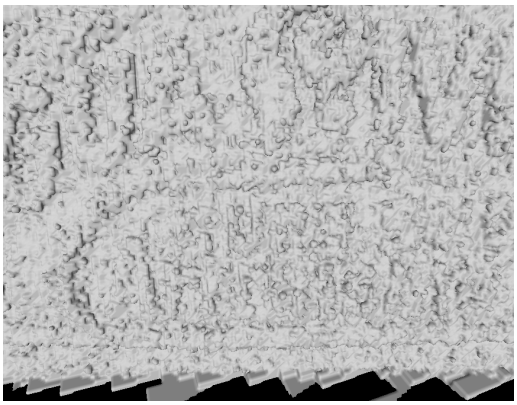
(c) 0,06 *mm* Poisson 9



(d) 0,11 *mm* Poisson 9



(e) 0,06 *mm Marching Cubes*



(f) 0,11 *mm Marching Cubes*

Fig. 4.28

Por otro lado en el escáner desarrollado se notó que el error de captura es localizado y no distribuido de forma uniforme. Si bien en cierta medida esto puede ser provocado por el error de fabricación de la placa, ya que se manifiesta en menor medida con el HandyScan 3D 300, esta localización de errores también indica que la calibración del sistema lograda, aunque exhaustiva, no fue la óptima, evidenciando lo dificultoso que puede llegar a ser lograr una óptima calibración del sistema.

A su vez, logramos comparar de forma exhaustiva el escáner desarrollado con uno industrial utilizado en la actualidad en entornos profesionales, llamado HandyScan 3D 300 y fabricado por Creaform, el cual es vendido en aproximadamente U$D40.000.

Logramos probar que nuestro escáner logra resultados similares, e incluso con más detalles que el HandyScan 3D 300 para algunas superficies. Hay que tener en cuenta que a pesar de que el escáner desarrollado en esta tesis obtiene resultados con calidad similares al HandyScan 3D 300, el tiempo de captura de nuestro escáner es significativamente mayor que el del HandyScan 3D 300. Esto sucede en el caso de una sola captura, pero empeora en el caso de necesitar varias capturas para cubrir el objeto, ya que no solo se necesitan realizar más capturas, sino que posteriormente se necesitan alinear dichas capturas, lo cual requiere de ayuda del usuario si se utiliza ICP desde una interfaz gráfica.

La amplia diferencia de tiempos de captura es principalmente producto de dos temas. Primero los requisitos de poder de cómputo exigidos por el HandyScan 3D 300 exceden ampliamente el poder de cómputo disponible en nuestro diseño, ya que incluso es necesaria una placa gráfica GPU y una PC poderosa para utilizarlo. A su vez el mismo dispositivo de escaneo realiza cómputos, y sincronización por hardware que permite una alta velocidad de adquisición. Segundo, la técnica de escaneo por láser implementada por el HandyScan 3D 300 junto a su alta velocidad de adquisición permiten un movimiento continuo del escáner que permite lograr una cobertura considerable del objeto a escanear desde múltiples ángulos de vista con comodidad, en menor tiempo y sin necesidad de reposicionar el objeto en muchos casos.

Con respecto a las técnicas de súper-resolución desarrolladas, se probó en primer lugar que la técnica basada en hardware utilizando la plataforma móvil logró incrementar la cobertura del objeto de escaneado sin reducir la precisión del sistema (Subsección 4.1.3), permitiendo generar nubes de puntos más densas y por lo tanto superficies de mejor calidad.

En el caso de la técnica de súper-resolución por *space carving* a pesar de lograr el tallado del objeto utilizando los rayos de cámara, es necesario desarrollar una mejor técnica de reconstrucción de superficie acorde al modelo muestreo generado, sin exhibir la partición del volumen tallado de forma explícita.

5. CONCLUSIONES Y TRABAJO FUTURO

5.1. Conclusiones

En esta tesis se diseñó e implementó un escáner 3D integrado de bajo costo basado en luz estructurada, de fácil uso y que logra capturar objetos 3D con una calidad equivalente a escáneres de uso profesional. La precisión de la nube de puntos obtenida por el escáner desarrollado es de al menos 0,041 mm. Mientras que la superficie generada a partir de dichos puntos logra una precisión de al menos 0,017 mm. Como referencia, el escáner profesional utilizado de forma comparativa obtuvo una precisión de al menos 0,024 mm.

Análogamente, la exactitud (métrica 2σ) del escáner es de al menos 0,082 mm para la nube de puntos y 0,044 mm para una superficie generada a partir de ellos, mientras que la exactitud alcanzada por el escáner comercial es de 0,059 mm certificada.

El escáner diseñado logra poner al alcance de equipos de investigación ajenos a la computación un escáner de alta precisión a un costo un orden de magnitud más baja que sus equivalentes comerciales. A su vez la interfaz desarrollada permite calibrar y realizar capturas 3D sin requerir conocimiento técnico específico.

El escáner a su vez integra una técnica de súper-resolución por hardware diseñada en esta misma tesis, que logra obtener una mayor cantidad de puntos 3D de la superficie escaneada que la normalmente obtenida utilizando captura estéreo con luz estructurada de forma habitual. Estos nuevos puntos 3D no son generados, sino que son capturados de la superficie.

A su vez se diseñó la una técnica de escaneo volumétrico por *space carving* que utiliza rayos de cámara para tallar un cubo 3D obteniendo un muestreo 3D del objeto escaneado. El muestreo resultante puede ser utilizado por múltiples algoritmos de reconstrucción de superficie como *Marching Cubes* y *Surface Nets*. Si bien el objetivo del desarrollo de esta técnica era lograr obtener súper-resolución por software, esto no logró superar la súper-resolución por hardware, abriendo una línea de trabajo futuro.

5.2. Trabajo futuro

5.2.1. Escáner 3D

Uno de los puntos débiles del escáner al compararlo con escáneres profesionales es el tiempo de captura. Si bien una limitación está dada por el poder de procesamiento de la computadora embebida, hay varias oportunidades de mejora en el diseño del software.

Por un lado se necesita mejorar el algoritmo que sincroniza la captura de imágenes con la proyección de patrones de luz. Para obtener mayor velocidad se pueden separar los controladores de las cámaras en procesos separados al proceso de escaneo principal. Esto permitiría tener en todo momento el framebuffer de las cámaras actualizado sin necesidad de disparar y esperar la una nueva captura en el momento que se necesita.

Por otro lado el algoritmo de renderización de patrones de luz estructurada podría implementar un buffer múltiple para incrementar la velocidad de proyección de patrones.

Otro cambio más estructural que lograría disminuir significativamente los tiempos de captura y procesamiento es la implementación en C o $C++$ del algoritmo de orquestación

de capturas, actualmente hecho en Python.

5.2.2. Súper-resolución por hardware

Una oportunidad de mejora a la técnica de súper-resolución por hardware desarrollada es calcular el tamaño de píxel del proyector de forma dinámica en el momento de captura. Si vemos la Figura 3.12, esto significaría que, en vez de calcular el tamaño del píxel del proyector proyectado usando el rango de trabajo del proyector (w) se podría calcular para la captura especifica. Esto se podría lograr, por ejemplo, utilizando la primera captura y midiendo la distancia promedio entre filas de píxeles del proyector, y dividiendo dicha distancia por la cantidad de capturas de súper-resolución a realizar ($cpart$).

5.2.3. Escaneo volumétrico por *space carving*

Para poder lograr súper-resolución a partir del escaneo volumétrico de un objeto es necesario en un principio diseñar o adaptar un algoritmo de reconstrucción de superficie de forma específica a la técnica desarrollada, que no se adapte explícitamente a la estructura del tallado generada.

Bibliografía

[ARV+14] Miguel Ares, Santiago Royo, Jordi Vidal, Laia Campderrós, David Panyella, Frederic Pérez, Sergio Vera, and Miguel A González Ballester. 3d scanning system for in-vivo imaging of human body. In *Fringe 2013*, pages 899–902. Springer, 2014.

[BAT13] Marijan Brozović, Andrej Avsec, and Marina Tevčić. Dimensional control of complex geometry objects using 3d scanning technology. In *14th International Scientific Conference on Production Engineering*, 2013.

[BM92] Paul Besl and H.D. McKay. A method for registration of 3-d shapes. ieee trans pattern anal mach intell. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14:239–256, 03 1992.

[Bra16] A.K.G. Bradski. *Learning OpenCV 3*. O'Reilly Media, Incorporated, 2016.

[BSK+19] Andreas Breitbarth, Timothy Schardt, Cosima Kind, Julia Brinkmann, Paul-Gerald Dittrich, and Gunther Notni. Measurement accuracy and dependence on external influences of the iphone x truedepth sensor. In *Photonics and Education in Measurement Science 2019*, volume 11144, page 1114407. International Society for Optics and Photonics, 2019.

[Cor13] Microsoft Corporation. Super-resolving depth map by moving pattern projector, 2013.

[DB09] Paul De Bièvre. The 2007 international vocabulary of metrology (vim), jcgm 200:2008 [iso/iec guide 99]: Meeting the need for intercontinentally understood concepts and their associated intercontinentally agreed terms. *Clinical biochemistry*, 42:246–8, 03 2009.

[FB81] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[Gib98] Sarah FF Gibson. Constrained elastic surface : Generating smooth surfaces from binary segmented data. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 888–898. Springer, 1998.

[GVS18] Silvio Giancola, Matteo Valenti, and Remo Sala. *A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopy Technologies*. Springer, 2018.

[HS97] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 1106–1112. IEEE, 1997.

[JWK+13] Mohammad Azam Javed, Seong-hoon Peter Won, Mir Behrad Khamesee, William W Melek, and William Owen. A laser scanning based reverse engineering system for 3d model generation. In *IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society*, pages 4334–4339. IEEE, 2013.

[KBH06] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[KH13] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.

[KS08] Avshalom Karasik and Uzy Smilansky. 3d scanning technology as a standard archaeological tool for pottery analysis: practice and theory. *Journal of Archaeological Science*, 35(5):1148–1168, 2008.

[KWFC16] M Kedzierski, D Wierzbicki, A Fryskowska, and B Chlebowska. Analysis of the possibilities of using low-cost scanning system in 3d modeling. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 41, 2016.

[LC87] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[LSS19] Free LSS. Free LSS. `http://www.freelss.org/`, 2019. [Online; accessed 03-november-2019].

[Luk15] Mario Lukas. *FabScan Pi-an open-hardware stand-alone web-enabled 3D scanner*. PhD thesis, RWTH Aachen University, 2015.

[MCSM18] Alexander Majercik, Cyril Crassin, Peter Shirley, and Morgan McGuire. A ray-box intersection algorithm and efficient dynamic voxel rendering. *Journal of Computer Graphics Techniques Vol*, 7(3), 2018.

[MT12] Daniel Moreno and Gabriel Taubin. Simple, accurate, and robust projector-camera calibration. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 464–471. IEEE, 2012.

[PCL+13] D. ProfessorPh., Ciobanu, D. LecturerPh., Chang Xing Xu, and D. The use of 3 d scanning and rapid prototyping in medical engineering. 2013.

[RBF+18] DF Redaelli, S Gonizzi Barsanti, P Fraschini, E Biffi, and G Colombo. Low-cost 3d devices and laser scanners comparison for the application in orthopedic centres. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(2), 2018.

[SJP13] Jan Smisek, Michal Jancosek, and Tomas Pajdla. 3d with kinect. In *Consumer depth cameras for computer vision*, pages 3–25. Springer, 2013.

[SKMK15] Jeremy Straub, Benjamin Kading, Atif Mohammad, and Scott Kerlin. Characterization of a large, low-cost 3d scanner. *Technologies*, 3(1):19–36, 2015.

[VGB] A Voicu, Gheorghe I Gheorghe, and L Badita. 3d measuring of complex automotive parts using video-laser scanning.

[WS16] Oliver Wasenmüller and Didier Stricker. Comparison of kinect v1 and v2 depth images in terms of accuracy and precision. In *Asian Conference on Computer Vision*, pages 34–45. Springer, 2016.

[WSRK11] Michael Weinmann, Christopher Schwartz, Roland Ruiters, and Reinhard Klein. A multi-camera, multi-projector super-resolution framework for structured light. In *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 397–404. IEEE, 2011.

[XA07] Yi Xu and Daniel G Aliaga. Robust pixel classification for 3d modeling with structured light. In *Proceedings of Graphics Interface 2007*, pages 233–240. ACM, 2007.

[Zha99] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 666–673. Ieee, 1999.