Jonatan Valencia González

Generación de dibujo paramétrico CAD, a partir de mallas de polígonos.

2026

Universidad Autónoma de Querétaro

Facultad de Ingeniería

Generación de dibujo paramétrico CAD, a partir de mallas de polígonos.

TESIS

Que como parte de los requisitos para obtener el Grado de

Maestro en Ciencias en Inteligencia Artificial

Presenta:
Jonatan Valencia González
Dirigido por:
Dr. András Takács

Universidad Autónoma de Querétaro

Facultad de Ingeniería

Maestría en Ciencias en Inteligencia Artificial

Generación de dibujo paramétrico CAD, a partir de mallas de polígonos.

TESIS

Que como parte de los requisitos para obtener el Grado de Maestro

en Ciencias en Inteligencia Artificial

Presenta:
Jonatan Valencia González
Dirigido por:
Dr. András Takács

Dr. András Takács
Presidente
Dr. Jesús Carlos Pedraza Ortega
Secretario
Dr. Juan Manuel Ramos Arreguín
Vocal
Dr. Gendry Alfonso Francia
Suplente
M.C. Luis Roberto García Noguez
Suplente

Centro Universitario, Querétaro, Qro.

Mayo de 2026

México

Dedicatoria

Los seres humanos hemos logrado un gran avance en gran medida gracias a nuestra capacidad de colaborar, sin duda alguna este trabajo no sería posible sin la colaboración y el apoyo de muchas personas, incluso de personas las cuales no conozco, ya que el impacto que tienen nuestras acciones puede ayudar o afectar a personas que incluso no existirán en nuestro mismo marco temporal. Seguramente habrá personas que se omitan en estos agradecimientos más por descuido que por intencionalidad.

Me gustaría iniciar agradeciendo a mis padres que con su esfuerzo y ejemplo aportaron siempre a mi desarrollo y a mí curiosidad, son definitivamente un pilar en mi vida y nunca podré regresarles lo mucho que me han dado. También quiero agregar a mi profesores y en especial a mi asesor el doctor András Takács, que creyó en mi proyecto y gracias a su voto de confianza fue posible la realización de mi proyecto.

También a mis compañeros de maestría y amigos que con su apoyo ideas y ánimo me ayudaron a lograr llevar a término está etapa.

También quiero hacer un agradecimiento general a las personas que de forma anónima o sin saberlo ayudaron a qué esto fuera posible.

Agradecimientos

Agradezco a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación por la beca con la cual se pudieron financiar los estudios y la presente investigación. Sin el apoyo de la secretaría este trabajo de investigación no hubiera sido posible. También agradezco a la Universidad Autónoma de Querétaro y a la dirección de Investigación y Posgrado de la Facultad de Ingeniería que permitieron realizar este trabajo, así como proporcionar las herramientas y estructura para llevarlo a cabo. Un especial reconocimiento a mi asesor de Tesis el Doctor András Takács, por creer en mi proyecto, y darme un voto de confianza para realizar esta investigación, así como de su constante guía y asesoramiento, sin su ayuda este trabajo no se hubiera realizado. Agradezco a los doctores Jesús Carlos Pedraza Ortega, y Juan Manuel Ramos Arreguín quienes ayudaron en mi formación con sus enseñanzas, ayudaron a darle forma al proyecto y a conseguir los entregables del mismo. Quiero agradecer también a todos los doctores que compartieron su conocimiento, tiempo y guía durante la maestría, ya que su esfuerzo fue fundamental en el proceso de formación. Por último quiero agradecer a mi madre y hermano por su constante apoyo, a mis amigos por siempre estar ahí, y a mi padre que hoy ya no está con nosotros pero sus enseñanzas me acompañan todos los días.

Abstract

3D mechanical design files are widely used, most of its characteristics are defined in the parametric drawing or sketch. In this work a method to reconstruct the sketch from contour images is proposed. Those images can be extracted from polygon mesh files. A synthetic database of contours images is generated, made only of straight lines, from 3 to 6 lines. Convolutional networks are used to process images and predict vertex coordinates. Then those coordinates are used to generate a script that is a code representation of the sketch, using the FreeCAD library in Python. Finally the script is executed in FreeCAD software, generating the sketch in the graphics user interface, which enables the full manipulation of the sketch.

Resumen

Los archivos 3D de diseño mecánico son ampliamente usados, la mayoría de sus características se definen en el dibujo paramétrico o sketch. Este trabajo propone un método para reconstruir el sketch a partir de imágenes de contornos que pueden ser generadas a partir de archivos de mallas de polígonos. Se genera una base de datos sintéticos, de imágenes de contornos, compuestos únicamente por líneas rectas, con entre 3 y 6 líneas. Se usan redes convolucionales para procesar las imágenes y predecir las coordenadas de los vértices de las líneas. Posteriormente con estas coordenadas se genera un script que es la representación en código del sketch, usando la librería de FreeCAD en Python. Finalmente el script es ejecutado en el software FreeCAD para generar el sketch en la interfaz gráfica de FreeCAD, lo que permite la completa edición y manipulación del sketch.

Abreviaturas y siglas

CAD: Diseño asistido por computadora

CAM: Manufactura asistida por computadora

IGES: Especificación inicial para el intercambio de gráficos

STEP: Estándar para el intercambio de datos del modelo del producto

STL: Lenguaje de triangulación estándar

MAE: Error medio absoluto

FEA: Análisis de elementos finitos

DS: Desviación Estándar

IoU: Intersección sobre union

PNG: Gráficos de Red Portátiles

ReLU: unidad lineal rectificada

GUI: Interfaz Grafica de Usuario

Índice General

Índice de Figuras

1. Introducción

Los archivos 3D son representaciones computarizadas de un objeto o pieza, estos archivos tienen una amplia gama de enfoques y aplicaciones. Los archivos 3D se pueden dividir en 2, de diseño mecánico y de diseño gráfico, las principales diferencias son el proceso de diseño, el tipo de geometría y la aplicación. Este trabajo se enfoca en archivos 3D de diseño mecánico. Ejemplos de las geometrías usadas en los diferentes enfoques de diseño se muestran en la figura 1.1.

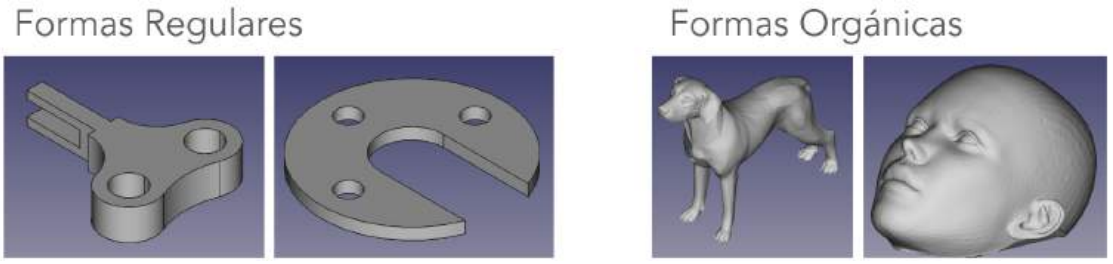


Figura 1.1. Diferentes tipos de geometrías objetivo en cada tipo de diseño CAD, las formas regulares son comúnmente usadas en el diseño mecánico, mientras que formas irregulares o orgánicas son usadas en el diseño gráfico.

El diseño gráfico usa formas irregulares las cuales se van esculpiendo y deformando, el proceso de diseño no se enfoca en dimensiones o parámetros, el diseño se centra en la geometría final y no en las operaciones que se usaron para crear la geometría, por lo que no hay forma de editar una operación pasada específica. Tiene aplicaciones en entretenimiento, mercadotecnia entre otras. Algunos de los inconvenientes con los objetos 3D de "diseño gráficos" es que no es fácil replicar geometrías, o especificar dimensiones, o características específicas, a su vez como no las operaciones que se esculpen no se guardan, no es posible editarlas, para cambiar sus parámetros.

Por otro lado en el diseño mecánico se usan geometrías regulares como líneas rectas y superficies planas. Las geometrías se controlan con parámetros de dimensión o restricciones geométricas. Se puede controlar por ejemplo el radio y punto tangente de un arco, o el ángulo y longitud de una línea. Un elemento como una línea recta no se puede deformar en una curva, hay que cambiar el tipo de geometría. El proceso de diseño se centra en ir definiendo geometrías y operaciones por etapas que se guardan en

un árbol de operaciones, por lo que cualquier operación pasada se puede editar. Todo esto permite diseñar geometrías muy complejas y editarlas de forma precisa para adaptarse a cambios, nuevas aplicaciones, corregir errores de diseño, entre otras. Estas cualidades hacen que los archivos 3D de “diseño mecánico” sean indispensables en aplicaciones como manufactura, ingeniería, análisis mecánico entre otras. Este trabajo se enfoca en estos archivos de diseño mecánico.

De forma simplificada para poder generar un "archivo 3D de diseño mecánico" se siguen 3 pasos que se muestran en la figura 1.2, el primero es seleccionar un plano, segundo generar un “dibujo paramétrico” o “sketch” sobre el plano. Este dibujo o “sketch” es el paso fundamental ya que es aquí donde se define la mayoría de las características del archivo 3D. El tercero y último paso es aplicar una “operación volumétrica” al dibujo paramétrico, puede ser por ejemplo una extrusión la cual agrega material, o un corte, el cual retira material. Repitiendo estos 3 pasos se crean geometrías complejas con variedad de aplicaciones.

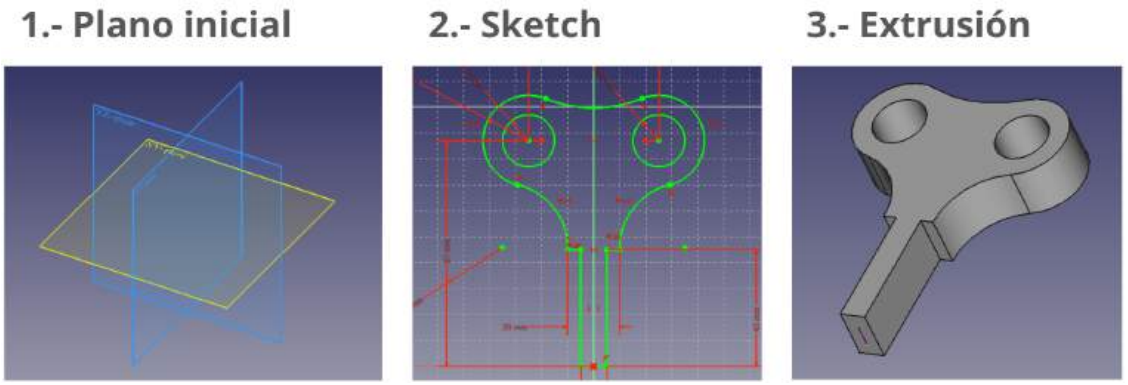


Figura 1.2. Pasos generales en el proceso de diseño mecánico.

El "sketch" o dibujo paramétrico es la base del archivo 3D de diseño mecánico, porque en el sketch se definen la mayoría de parámetros que definen la geometría de un archivo 3D. Estos parámetros son: geometrías primitivas (por ejemplo líneas y arcos), geometrías paramétricas (curvas, trayectorias, patrones), restricciones dimensionales (longitud, radio), restricciones geométricas (tangencia, perpendicularidad, simetría), referencias a otras geometrías (por ejemplo para ensamble).

Las operaciones y parámetros descritos anteriormente se guardan en un "árbol de operaciones", se almacenan: los "sketch", las operaciones volumétricas, geometrías de apoyo, o alguna otra información como el tipo de material. Este árbol de operaciones es muy útil, ya que nos permite tener una especie de histórico o de registro de todo lo que fue realizando para modelar o esculpir la pieza 3D, por ejemplo en la figura 1.3 vemos a la izquierda este árbol de operaciones, en este caso particular el archivo se llama "ex08", de aquí se desprende el árbol de operaciones donde podemos ver dos "operaciones volumétricas" una que se llama "pad" o extrusión, y otra que se llama "pocket" o recorte. De estás dos operaciones se desprende los dos "sketch" que las originan, en este caso se llaman "sketch" y "sketch001", recordemos que en estos "sketch" o dibujos paramétricos es donde se definen la mayoría de geometrías.

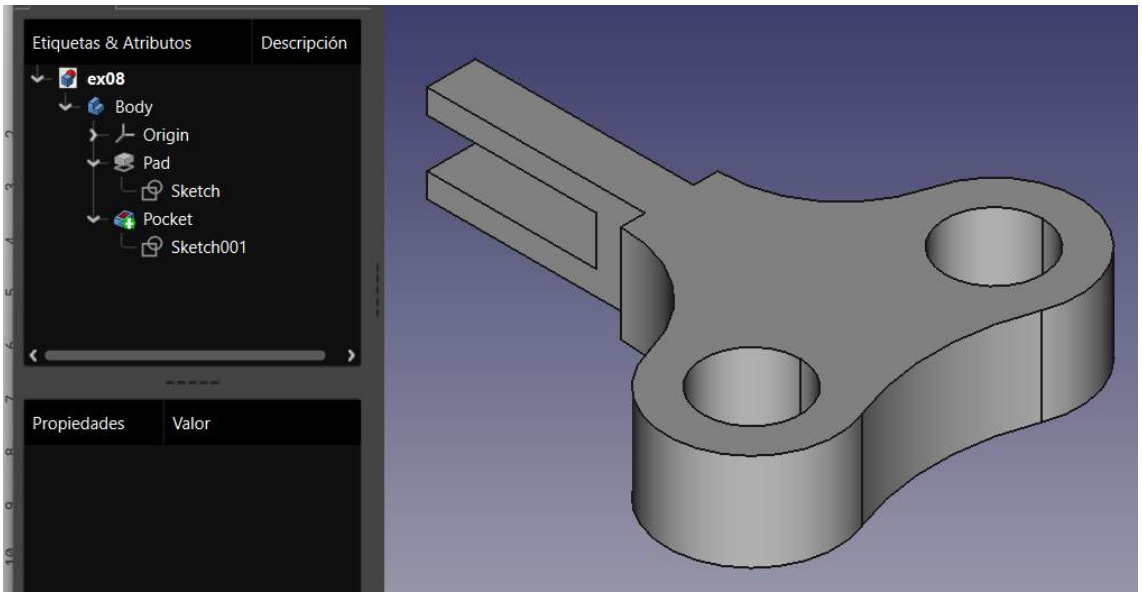


Figura 1.3. Árbol de operaciones en el software de diseño mecánico FreeCAD.

En este caso la pieza es sencilla y solo tienes dos operaciones volumétricas con sus dos sketch respectivos, pero en la mayoría de piezas o diseños de la vida real se agregan más operaciones, y otros detalles menores, por ejemplo "redondeos" o "chaflanes" según sea el caso. También en esta geometría la "características internas" es decir los agujeros están definidos en un solo sketch, pero como comentario, siguiendo las prácticas de diseño se recomienda que las geometrías internas se definen en un "sketch" separado, para poder tener un control separa por ejemplo en caso de que aplicara

cambiar el tipo de agujero para agregar por ejemplo la definición de una rosca. Aquí por ejemplo la parte del "pocket" define el "corte" en la parte de la llave. De nueva cuenta un diseño más elaborado tendrá más operaciones volumétricas, con sus respectivos sketch, por ejemplo los agujeros o "características internas".

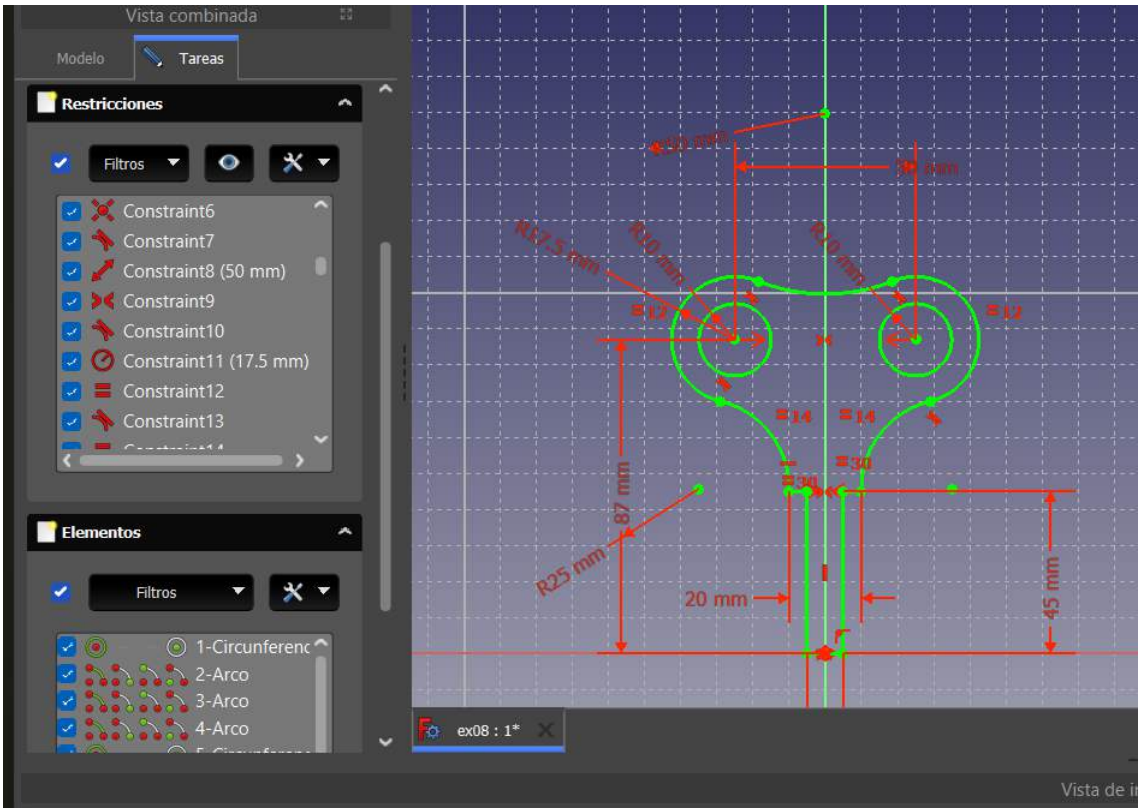


Figura 1.4. Elementos del sketch, como geometrías, restricciones geométricas, y restricciones dimensionales.

Dentro del "sketch" como se muestra en la Figura 1.4, podemos observar que en la parte izquierda, existen dos secciones, "restricciones" y "elementos". Estas dos secciones contienen la información del sketch, dentro del apartado de "restricciones" se encuentran las restricciones geométricas, y las restricciones dimensionales.

Las restricciones geométricas son relaciones entre geometrías por ejemplo una línea es horizontal respecto a los ejes de coordenadas, o una curva es tangente a una línea recta, un punto es coincidente con otro, entre otras relaciones.

En el caso de las restricciones dimensionales, son restricciones que limitan el tamaño de alguna geometría por ejemplo: las longitud de una línea recta, el radio de un arco, la distancia entre dos líneas, el ángulo entre dos líneas, entre otras restricciones dimensionales.

En el apartado de "elementos" tenemos las geometrías usadas en el "sketch", por ejemplo un arco, una línea recta, un círculo, entre otras.

Estos elementos dentro del sketch se pueden eliminar modificar o agregar nuevos en caso que sea necesario, y como ya se mencionó esta información se va guardando en el árbol de operaciones lo que permite tener una especie de histórico del proceso de diseño que un objeto 3D, También podemos observar que en el "sketch" es donde se especifica la mayoría de información de la geometría, es decir que especifican elementos geométricos, restricciones dimensionales, y restricciones geométricas, lo que permite diseñar geometrías muy complejas, con un alto grado de precisión y de repetibilidad, lo que lo hace ideal para una gran cantidad de aplicaciones como análisis técnicos, manufactura, prototipos entre otras.

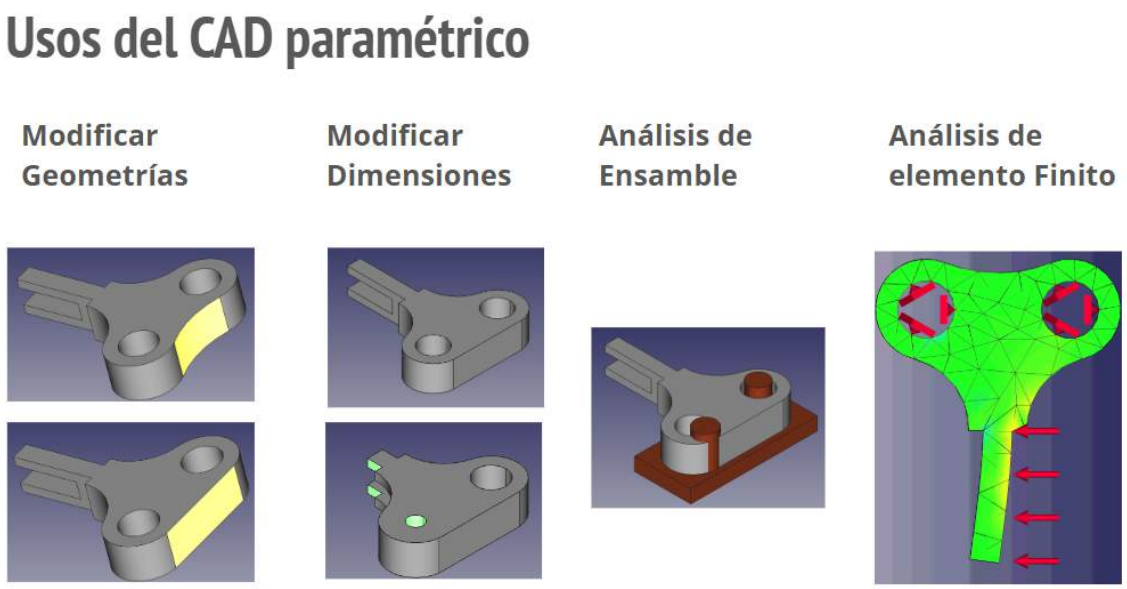


Figura 1.5. Diferentes aplicaciones de los archivos "paramétricos" o de diseño mecánico. En la imagen se observan de izquierda a derecha, la modificación de geometrías, modificación de dimensiones, análisis de ensamble y de elemento finito.

Los "archivos 3D de diseño mecánico" son ampliamente usados en manufactura, ya que la gran mayoría de los objetos que se producen en la actualidad requieren un archivo 3D de diseño mecánico, ya sea para diseñar la pieza que se va a fabricar como un engrane o pistón, o para diseñar la herramienta que va a fabricar la pieza, como moldes de inyección para la suela de un tenis. Como se muestra en la figura 1.5, estos archivos tienen una amplia gama de aplicaciones.

Estos archivos de diseño mecánico tienen algunas limitaciones, por ejemplo: compatibilidad entre diferentes software de diseño, compatibilidad entre versiones de un mismo tipo de software, poca disponibilidad de los archivos "fuente", entre otras.

Existen formatos que permiten tener compatibilidad entre diferentes software de diseño y sus versiones, uno de ellos es el: "Lenguaje de Triangulación estándar" (STL por sus siglas en inglés).

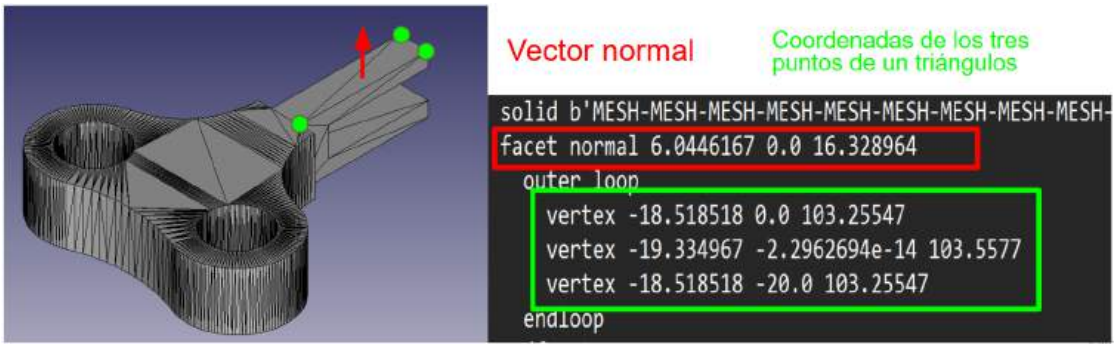


Figura 1.6. Estructura de un archivo STL. En verde se muestran las coordenadas espaciales de los vértices. En rojo se muestra el vector normal a la cara del triángulo, lo cual indica el lado exterior del objeto 3D.

El formato STL de forma genérica se puede clasificar como un archivo de "mallas de polígonos". Es un tipo de archivo simplificado en el cual el objeto 3D se representa al describir polígonos que recubren la superficie exterior del objeto 3D. Es uno de los formatos más ampliamente usado, por ejemplo se usa para compartir diseño en manufactura, para realizar impresiones 3D entre otras aplicaciones.

En el formato STL solo se usan triángulos, el archivo consta de una lista de triángulos no conectados entre sí. Los datos que describen estos triángulos son un vector normal a la superficie del triángulo y las coordenadas espaciales de los 3 vértices. El vector

normal se usa para definir la cara exterior del objeto 3D y los vectores para definir el triángulo.

En la figura 1.6 se muestra la estructura del formato STL. Se puede observar en verde marcadas las coordenadas de los tres vértices de uno solo de los triángulos, así como en rojo el vector normal a la superficie creada por ese triángulo, y que apunta hacía el exterior del objeto 3D.

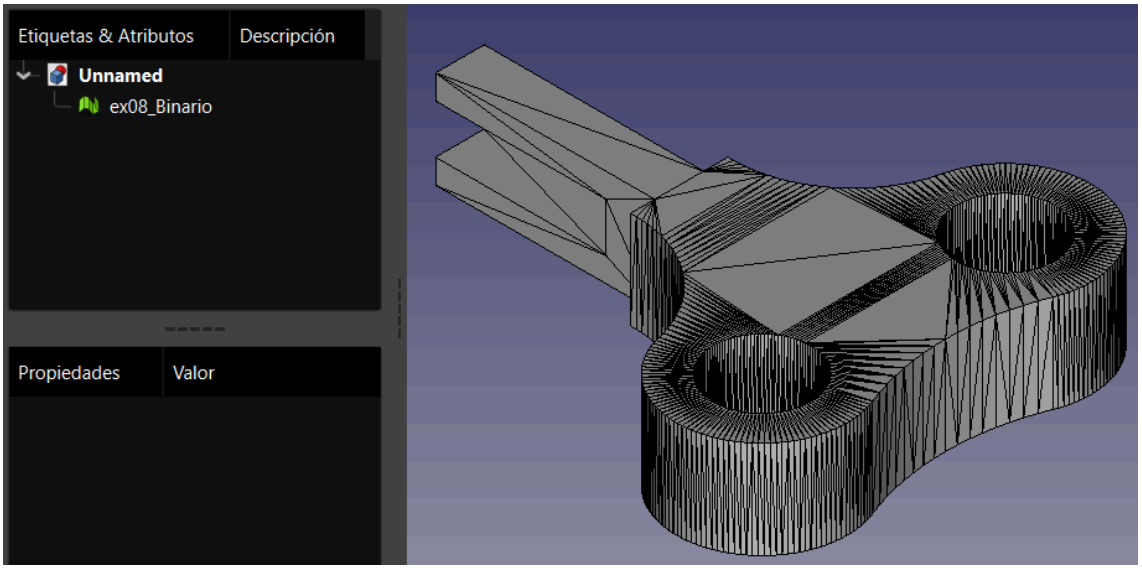


Figura 1.7. Árbol de operaciones de un archivo de mallas de polígonos en formato STL, donde se aprecia exclusivamente la malla sin más información de diseño.

Como se mencionará más adelante este tipo de archivos de mallas de polígonos tienen el problema que solo muestran la geometría final pero se pierde la información de diseño. En el caso del árbol de operaciones del archivo STL en lugar de tener las operaciones que lo generan únicamente es una malla, o malla de polígonos, recordando que estos polígonos son exclusivamente triángulos.

Al visualizar el archivo STL, en la interfaz gráfica de FreeCAD, se puede observar en el "árbol de operaciones" que únicamente se visualiza un "elemento", este elemento es la malla de polígonos, esto se puede observar en la parte izquierda de la figura 1.7.

Podemos observar también la característica de la malla ya que la geometría está rayada en otras palabras está recubierta por triángulos que forman la malla. En este caso esa malla de polígonos, se visualiza en el árbol de operaciones en la parte izquierda.

2. Justificación

Los archivos de diseño 3D de diseño mecánico, tiene una amplia gama de aplicaciones en ingeniería, manufactura, diseño, varios tipos de análisis, creación de refacciones entre otros. Estos archivos no están disponibles para los usuarios generales. La reconstrucción de los archivos 3D es costosa y técnicamente complicada, no existe compatibilidad entre diferentes marcas de software de diseño mecánico y en ocasiones no hay compatibilidad entre versiones del mismo software. Por estas limitaciones y ventajas, la capacidad de construir los archivos 3D de diseño mecánico de forma automática abriría el acceso a la amplia gama de aplicaciones antes mencionada, reduciendo costos, y necesidad de conocimiento técnico.

3. Descripción del problema

Si bien los archivos 3D de diseño mecánico son ampliamente usados, estos archivos rara vez están disponibles, es decir solamente los tiene la empresa que fabricó el objeto originalmente esto limita la capacidad de edición, modificación y la capacidad de volver a fabricar las piezas sobretodo pensando en equipo especializado o descontinuado del cual no se puedan conseguir refacciones.

Incluso si se cuenta con el archivo original para poder usarlo sería necesario el software con el que fue creado, ya que no hay compatibilidad entre diferentes marcas de software, y en ocasiones no hay compatibilidad entre versiones del mismo software.

El método tradicional para crear el archivo sería tomar medidas y realizar ingeniería inversa para ir reconstruyendo el archivo 3D, esto implica altos costos, como el tiempo, la habilidad técnica especializada de diseño, el conocimiento de metrología para medir

la pieza, entre otros, lo cual hace muy complicado reconstruir el archivo 3D, limitando las aplicaciones.

A causa de estas limitaciones, y considerando las ventajas mencionadas anteriormente como la capacidad de edición y modificación, poder reconstruir de forma automática los "archivos 3D de diseño mecánico" abriría una amplia gama de posibilidades, por las aplicaciones que tienen: como en análisis de ingeniería, análisis de elemento finito, de tolerancias, análisis de ensamble, modificación de geometrías, reparación de errores de diseño, creación de accesorios para la pieza analizada, reconstrucción de refacciones entre otros.

El problema de compatibilidad se puede resolver de cierta forma usando archivos de mallas de polígono, por ejemplo en formato STL.

Figura 3.1. Comparación del árbol de operaciones entre un archivo de malla de polígonos y un archivo CAD o de "diseño mecánico".

En la figura 3.1 podemos ver una comparación entre el formato CAD o de diseño mecánico, y el formato STL. Al comparar el árbol de operaciones de ambos formatos vemos que en el caso del archivo CAD, se muestran los sketch, y operaciones volumétricas que se realizaron durante el proceso de diseño mientras que en el caso de archivo STL solo se aprecia una malla.

En el archivo STL en el árbol de operaciones se puede ver solo una malla de polígonos, la cual se llama "ex08_Binario", también podemos observar en la parte derecha el objeto 3D el cual está "mallado" o recubierto de triángulos que forman la malla.

El problema con el formato STL es que solamente es la malla pero no tiene más operaciones, por ejemplo no se guarda la información del sketch, o las operaciones volumétricas, es por esta falta de información que el alcance de estos archivos de mallas de polígonos está limitada.

Por ejemplo supongamos que deseamos modificar el diámetro de los agujeros, o que se desea añadir o modificar alguna geometría por ejemplo quitar uno de los redondeos, o modificar el radio de alguna geometría curva. En estos casos estas modificaciones son imposibles ya que no hay información o "objetos" que se puedan modificar.

Otra de las limitaciones de los archivos STL es que no se pueden realizar análisis con estos archivos por ejemplo mallado finos para "análisis de elemento finito" o para algún análisis de "esfuerzo deformación" o algún análisis de ensamble por ejemplo para encontrar las tolerancias que puedan funcionar en toda la cadena de ensamble en las diferentes piezas.

4. Antecedentes

Si bien hay varios trabajos enfocados en modificar y procesar archivos 3D de diseño mecánico, estas "modificaciones" parte de una gran variedad de punto de inicio, y con una gran variedad de objetivos finales.

Uno de los puntos claves de este trabajo es la "reconstrucción del sketch" o dibujo paramétrico. En este aspecto ningún artículo se ha enfocado en la reconstrucción del sketch, los que más se acercan hacen una descripción con un "lenguaje de dominio

específico". Este lenguaje es propietario y no permite su modificación ni ejecución en algún software de código abierto como FreeCAD.

Otro punto clave de este trabajo es que se parte de objetos no paramétricos, como lo son archivos de malla de polígonos o imágenes. Existen otros trabajos que también toman estos tipos de datos como entrada pero ninguna se acerca a la reconstrucción del sketch.

A continuación se discuten trabajos similares de forma resumida pero más adelante se muestra una descripción más detallada. Los antecedentes se pueden dividir en tres categorías, enfocados en modificación de archivos de malla de polígonos, en específico en formato de lenguaje de triangulación estándar (STL por sus siglas en inglés), trabajos enfocados en el procesamiento de los archivos 3D y sketch, y por último, enfocados en la identificación de vértices.

4.1. Resumen de antecedentes

Respecto a la modificación de archivos de malla de polígonos en formato STL tenemos, el caso de [1] que busca mejorar el mallado del archivo para realizar análisis de elemento finito, en el trabajo presentado se puede realizar una mallado con el archivo CAD generado por el sketch reconstruido o el caso de [2] se intentan modificar los STL mediante operaciones booleanas lo cual permite modificar los archivos 3D pero de forma limitada. El sketch al ser paramétrico se puede modificar en su totalidad.

También tenemos el caso de [3] que busca generar una malla de control para modificar el STL, de nueva cuenta esto permite solo cierto nivel de control de forma limitada. En [4] busca identificar superficies en el archivo de mallas de polígonos pero no avanza a generar el sketch.

Respecto al procesamiento del archivo 3D y sketch tenemos los siguientes trabajos, el caso de [5] que busca analizar el sketch mediante la representación de grafos, el problema de este trabajo es que no genera el sketch y solo analiza sus partes, también tenemos el caso de [6] y [7] que busca generar una representación del sketch pero mediante un lenguaje de dominio específico el cual es propietario lo cual limita su uso, además parten de modelos 3D ya existentes.

Siguiendo con trabajos de reconstrucción del sketch, en [8] se busca digitalizar dibujos a mano al identificar sus elementos, pero no llega a generar el sketch como tal en una interfaz gráfica enfocado en su modificación y parametrización, por lo que su uso es limitado.

Parte del enfoque de este proyecto es reconstruir el sketch desde imágenes para lo cual es útil identificar vértices en imagen. por eso ahora revisamos trabajos que busquen encontrar bordes o vértices en imágenes.

El caso de [9], busca encontrar los bordes y por último vértices para describir los polígonos para poder encontrar edificios en imágenes satelitales, su método toma como entrada el mapa de probabilidad de una red convolucional para después procesarlo, y encontrar los vértices. En [10] se extraen los vértices desde una imagen usando una red convolucional para generar un mapa de características, posteriormente crear una máscara de detección al pasar el mapa de características por una capa convolucional 1x1, y posteriormente extraer los vértices con un algoritmo de supresión de "no-máximos", para extraer los picos más relevantes que serían los vértices. El caso de [11] para encontrar los vértices o en este caso esquinas, usa árboles de decisión, la idea es tomar un píxel "x" que se va comparar con los pixeles que están a su alrededor, y ver si son más oscuros, más brillantes o similares. Con esta información luego se calcula la entropía para clasificar si el píxel "x" que se está analizando es un vértice o no.

En la identificación de polígonos para edificios en imágenes satelitales, en [12] usan una red convolucional con forma de U también llamada "U-net" para generar 3 salidas, una máscara de bordes, una máscara interior, y un "campo de marcos", estas salidas de la U-net son procesadas posteriormente para generar los polígonos de edificios, en específico la parte de encontrar los vértices se realiza un procesamiento posterior, el cual consiste en alinear la máscara interior con el "campo de marcos", para detectar los bordes usando el "modelo de esqueleto activo", y por último se realiza la detección de los bordes usando "campos de marcos" eliminando los vértices que no está en esquinas.

Las propuestas anteriores parten de puntos de inicio diferentes al propuesto en este trabajo, o no llegan a generar el sketch o dibujo paramétrico en la interfaz de usuario gráfica lo que limita las aplicaciones y modificación de geometrías.

4.2. Modificación de mallas de polígonos

Existen varios trabajos que estudian o abordan de cierta manera los archivos de mallas de polígonos, recordando que el enfoque de este trabajo es reconstruir el "sketch", y en consecuencia "modificar" los archivos 3D. En este apartado nos vamos a centrar en trabajos que buscan modificar o controlar los archivos de malla de polígonos, ya sea para modificar geometrías, o para expandir su aplicación, por ejemplo para mallado de de elemento finito.

En el caso del trabajo [1] tenemos un acercamiento que busca mejorar la "malla" para realizar "análisis de elemento finito" y así expandir los alcances del archivo de mallas de polígonos. En el caso del análisis de elemento finito se requiere que la malla tenga algunos aspectos, por ejemplo: que el tamaño de los polígonos sea aproximadamente similar, que esté distribuida de forma uniforme, entre otras.

Podemos ver en la Figura 4.1, un objeto 3D en formato STL, el cual está "poblado" por triángulos, se puede notar que hay zonas donde el tamaño de los triángulos es mayor que en otras, esto es propio del formato STL y cumple la función de simplificar el modelo, pero no es lo mejor para el caso del "análisis de elemento finito".

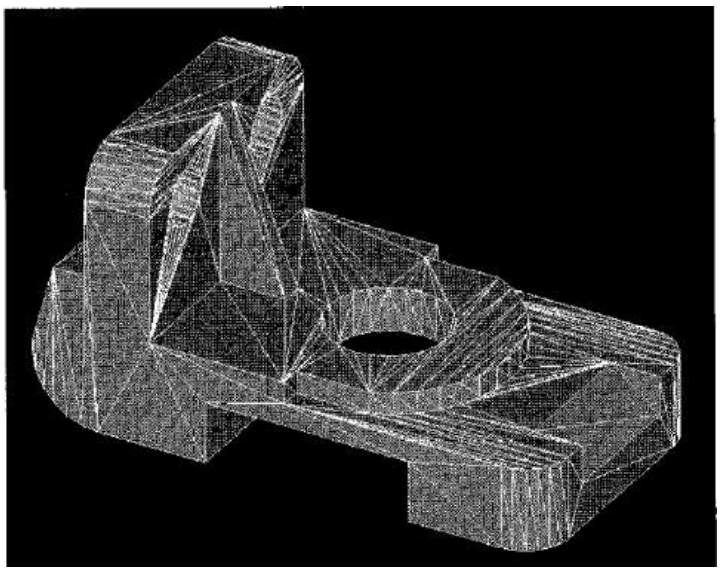


Figura 4.1. Objeto 3D mallado en formato STL, recuperado de [1].

En la figura 4.2, podemos ver que el modelo está mallado de forma más “fina” esto permite realizar el análisis de elemento finito. De nueva cuenta se puede apreciar que la malla es más pequeña, y aproximadamente de tamaño similar.

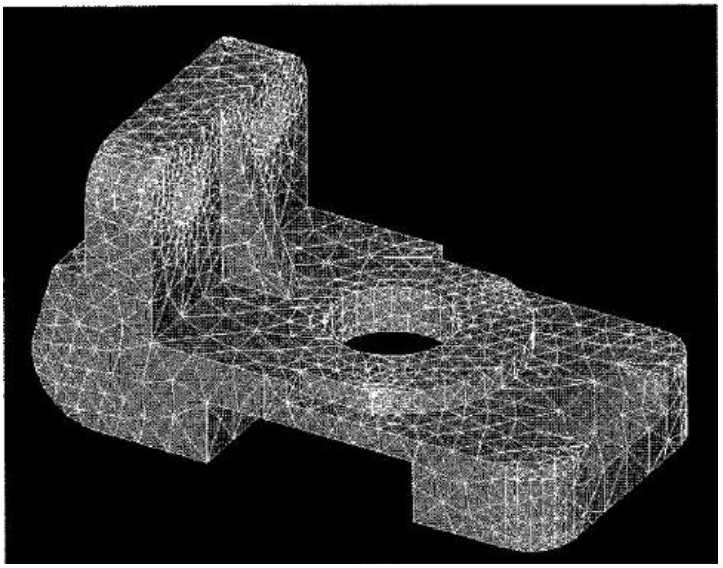


Figura 4.2. Objeto 3D con malla fina, recuperado de [1].

El objetivo del “análisis de elemento finito” es simular cómo se comportaría una geometría o objeto 3D en la vida real en condiciones como aplicaciones de fuerza, distribución de temperaturas entre otras. Con esto en mente la idea es “simplificar” el objeto 3D en partes más pequeñas para simular cómo se comporta en la vida real. Mientras más pequeña sea la malla mejor será la simulación, pero será más complicado simularlo por la carga computacional. Esta opción de ampliar la capacidad de un STL es interesante pero está limitada, y no permite por ejemplo editar la geometría.

Ahora pasamos al caso de editar la geometría de un archivo STL, tenemos el caso del trabajo descrito en [2] donde tomando dos geometrías se busca realizar operaciones booleanas para realizar ediciones rápidas. La idea consiste en traslapar dos objetos 3D en formato STL, luego analizar cuáles son los “triángulos” de cada una de los dos objetos que se interseccionan, para con esto crear un “aro” de “segmentos de intersección” con lo cuales se va a generar la nueva malla.

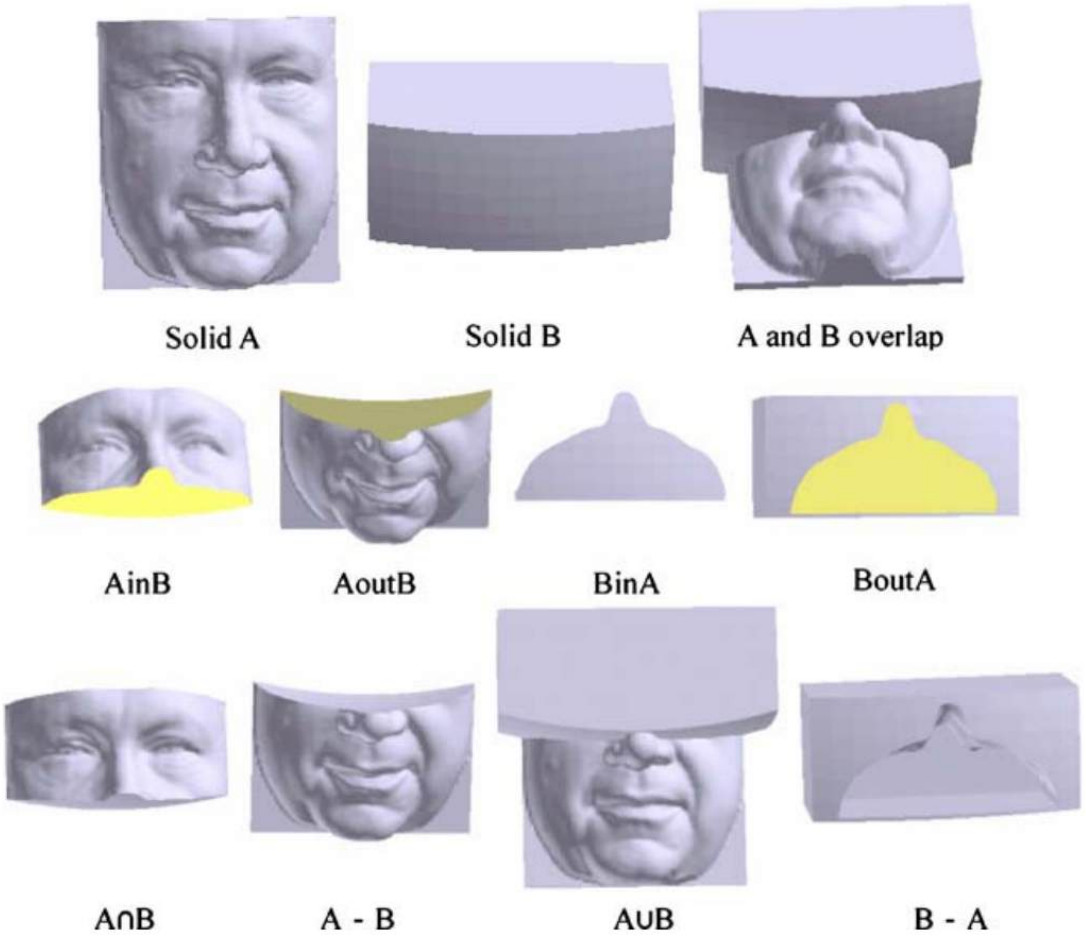


Figura 4.3. Operaciones Booleanas entre dos archivos STL. Recuperado de [2].

En la figura 4.3 vemos varias operaciones booleanas, toma como punto de partida dos sólidos los cuales son A y B. Se observa que el sólido A es un rostro humano, mientras que B es un volumen geométrico simple. Con estos dos sólidos podemos realizar varias operaciones lo cual a su vez tiene varias posibles salidas. Por un lado podemos unir, o sobreponer ambos sólidos, y tener una especie de suma de los objetos 3D. Por otro lado tenemos la operación de todo lo del cuerpo A que esté dentro de B y en ese caso se obtiene una especie de máscara. En otra operación se puede retirar todo el espacio que conocida con B y de esta forma quedarnos solo con alguna sección de interés.

El problema con las operaciones booleanas es que no nos permite editar de forma "libre" la geometría y tampoco se pueden modificar los parámetros, ya que el objeto de

salida sigue siendo un archivo STL y tampoco es posible usar el archivo para análisis u otros objetivos.

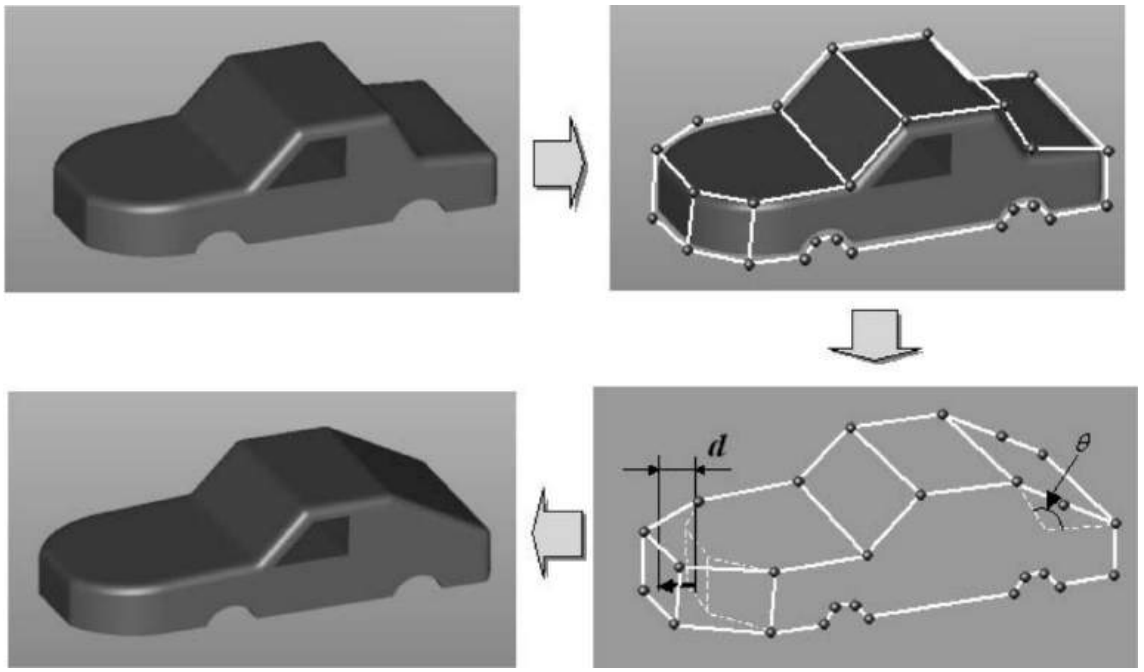


Figura 4.4. Control paramétrico con malla externa. En las dos imágenes de la parte superior vemos el archivo "original" en la parte superior izquierda, en la parte superior derecha la malla de control "original". En las imágenes inferiores vemos del lado derecho la malla de control agregando una distancia "d" o un ángulo "Θ", y del lado inferior izquierdo el resultado. Recuperado de [3].

Siguiendo en el tema de editado de archivos STL pasamos a un acercamiento de controlar el objeto 3D con parámetros, es el caso del trabajo descrito en [3], donde usando una "malla de control externa", se pueden agregar parámetros para controlar o modificar la el objeto 3D o "malla" original. Cómo se observa en la figura 4.4 donde vemos que el frente del vehículo se hace más "largo", y su parte posterior cambia de ser recta a ser diagonal.

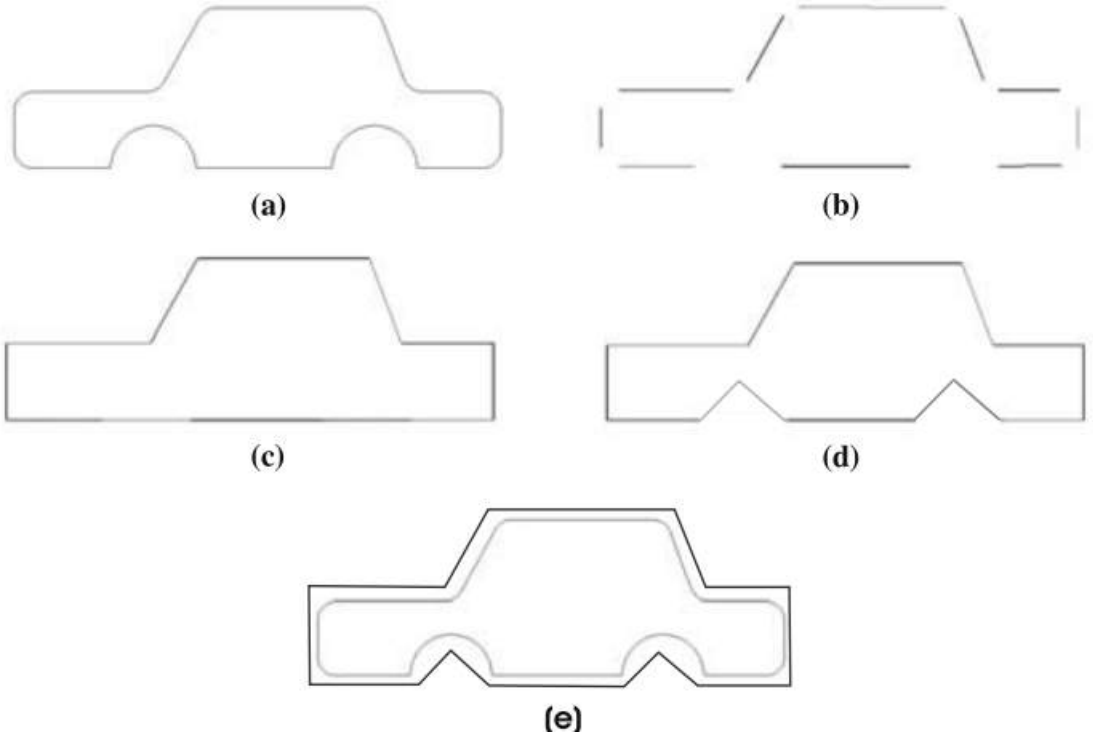


Figura 4.5. Procedimiento para calcular en polígono de control 2D. (a) Contorno original. (b) Detección de líneas. (c) Polígono cerrado. (d) Procesar el área de curvas. (e) Polígono de control con desfase. Recuperado de [3].

La idea para generar la malla de control externa es la siguiente: 1.- Extraer el contorno de una "cara", 2.- Simplificar el contorno en línea rectas o identificar líneas rectas simples 3.- Unir estas líneas en un polígono cerrado 4.- Procesar y agregar detalles de "curvas" 5.- Generar el polígono de control con un desfase. Este proceso se puede observar en la figura 4.5.

El problema con las mallas de control es que si bien se pueden modificar de forma "paramétrica" este cambio se hace "deformando" la geometría por lo que otras geometría que no se deseen modificar también pueden ser afectadas, y de igual forma el archivo final sigue siendo un STL por lo que se limita su aplicación para otros propósitos.

Ahora pasamos a la parte de reconocer "características" geométricas en los archivos de mallas de polígonos, recordando que los archivos STL son solamente una lista de los

triángulos que recubren la superficie exterior del objeto 3D. El objetivo del trabajo descrito en [4] es identificar "características" geométricas por ejemplo, caras planas, agujeros, redondos, entre otras.

Tabla 4.1. Tipo de geometría por propiedades de curvatura. Recuperado de [4].

Tipo de Superficie	Curvatura de Gauss (K)	Curvatura media (H)
Pico (peak)	+	-
fosa (pit)	+	+
cresta (ridge)	**0**	-
valle (valley)	**0**	+
Silla de cresta (Saddle ridge)	-	-
Silla de valle (Saddle valley)	-	+
Superficie mínima (Minimal Surface)	-	**0**
Plano (flat)	**0**	**0**

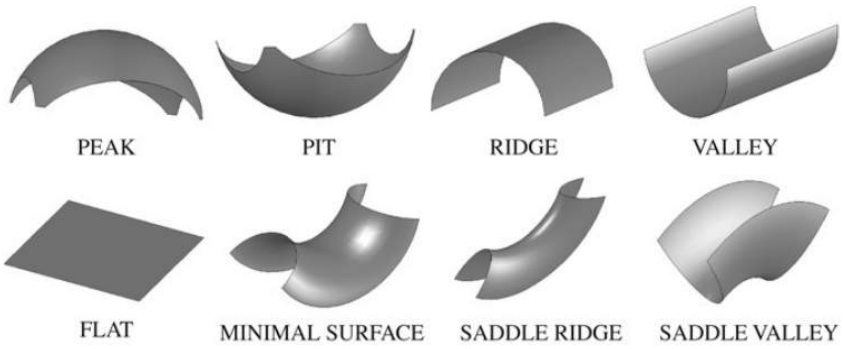


Figura 4.6. Formas fundamentales según sus curvas de Gauss y curva media. Recuperado de [4].

Para identificar estas "características" geométricas se hace un análisis de los ángulos entre los triángulos, los cuales pueden resultar entre 8 casos que se muestra en la tabla 4.1 y en la figura 4.6. Aquí se muestran 8 casos fundamentales los cuales son independiente de su ubicación o rotaciones ya que se analizan con las relaciones entre los triángulos subyacentes.

Con estas “características” geométricas básicas se pueden a su vez combinar en geometrías más complejas. También se puede interpretar como que geometrías más complicadas se pueden describir geometrías simples.

De cierta forma lo que se busca lograr al identificar geometrías más sencillas, es describir los bloques “mínimos” necesarios para describir geometrías 3D.

Se puede entender como las letras del abecedario que sirven como ingredientes para generar una receta más compleja que en ese caso sería una palabra u oración.

Otra forma de verlo sería con bloques de piezas de lego las cuales se pueden unir en diversas formas.

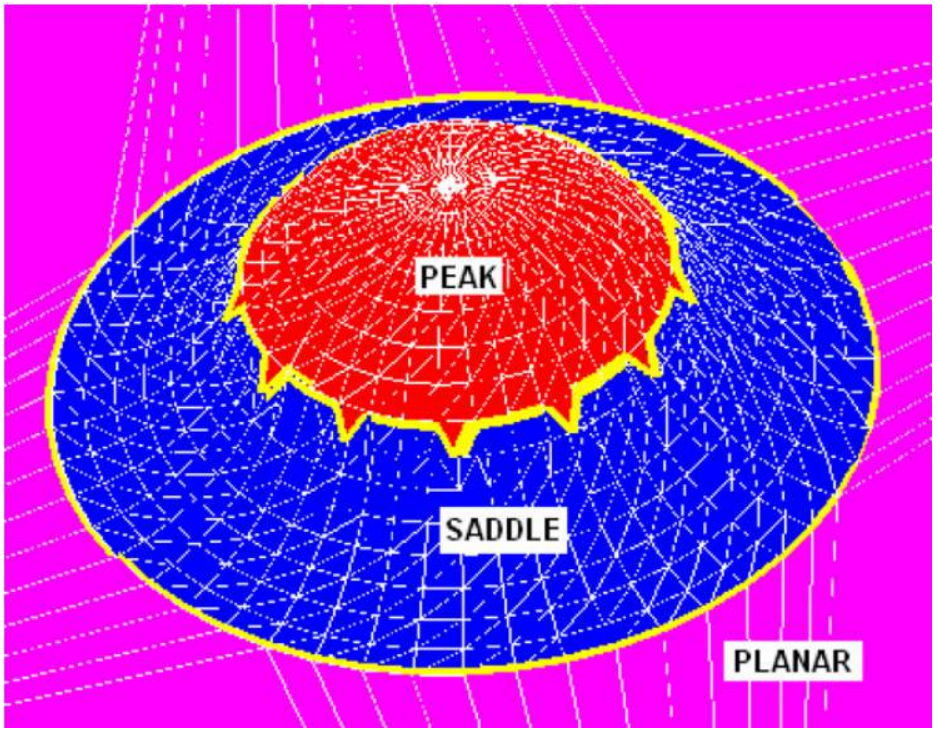


Figura 4.7. Geometría compleja de “abultamiento”. Recuperado de [4].

En la figura 4.7 vemos como una geometría más compleja como lo puede ser un “abultamiento” o “montaña”, se puede describir con tres de la forma fundamentales, en rosa podemos ver la zona o “característica” plana, en azul se observa el redondeo, y en rojo el “pico” o media esfera.

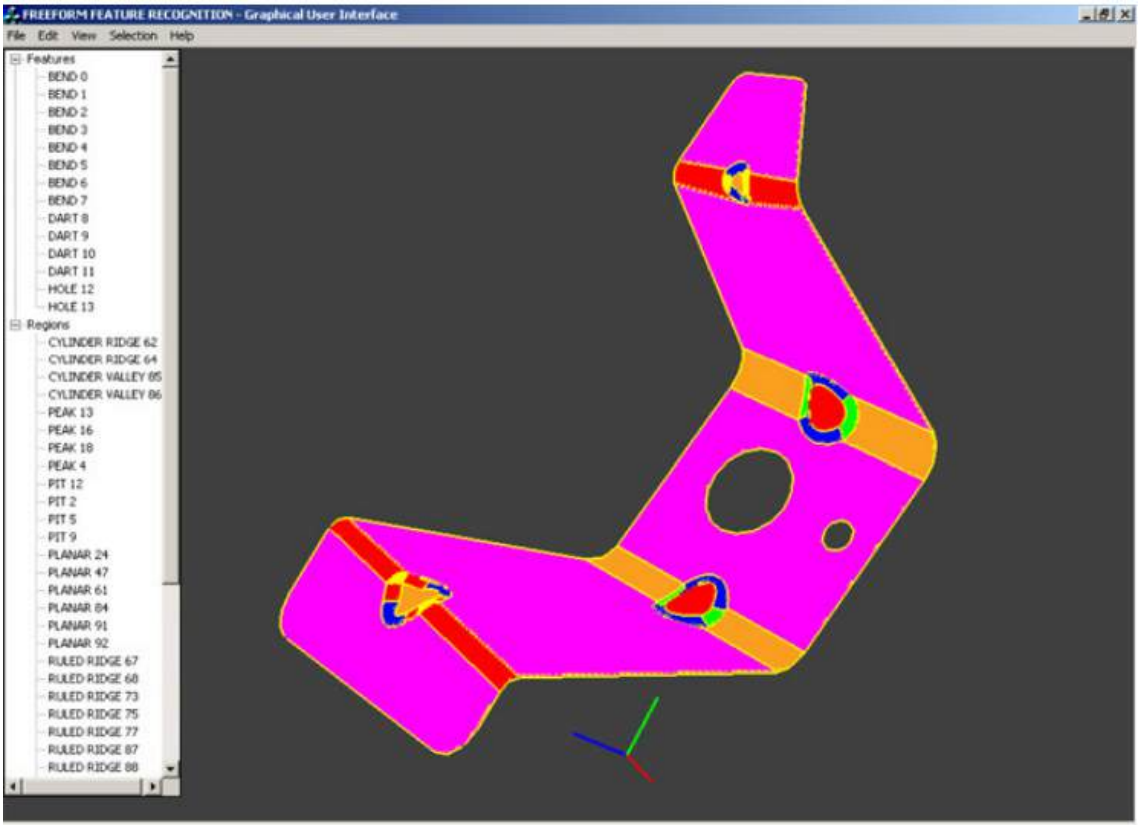


Figura 4.8. Geometría de un objeto 3D completamente identificada según las "formas fundamentales". Recuperado de [4].

Ahora repitiendo este análisis se puede aplicar en toda la geometría para identificar sus "características" lo que se puede observar en la figura 4.8. Donde podemos ver como toda la geometría ha sido identificada según sus formas fundamentales.

Cabe resaltar que este análisis se aplicó a formas se "lamina metálica" en la cual por el tipo de diseño ya que la materia prima son láminas podemos encontrar una gran cantidad de zonas planas así como una mayoría de redondeos. Esto puede limitar su aplicación en geometría más complejas, o con otro enfoque de diseño.

A forma de resumen en este apartado que se enfoca en trabajar con archivos de malla de polígonos en formato STL, podemos ver que sus puntos debiles en comparacion con el trabajo presentado son que, no se busca generar el dibujo paramétrico, que no se

reconstruye como tal un archivo de diseño mecánico y que los cambios se quedan exclusivamente en archivo de malla de polígonos en formato STL. Esto limita las modificaciones, uso en otras aplicaciones, y la capacidad en general del archivo.

4.3. Archivos 3D y Sketch

En este apartado de los antecedentes nos vamos a enfocar en trabajos que aborden o trabajen con archivos 3D como datos de entrada, o que de alguna forma aborden o trabajen con el dibujo paramétrico o sketch.

En el trabajo [5] se busca generar varios sketch de forma automática, esto se intenta lograr mediante la representación del sketch como grafos, donde la geometría primitiva sería el nodo, y las restricciones geométricas serían las aristas. Ahora para poder generar estos grafos se propone generar una sintaxis o gramática particular.

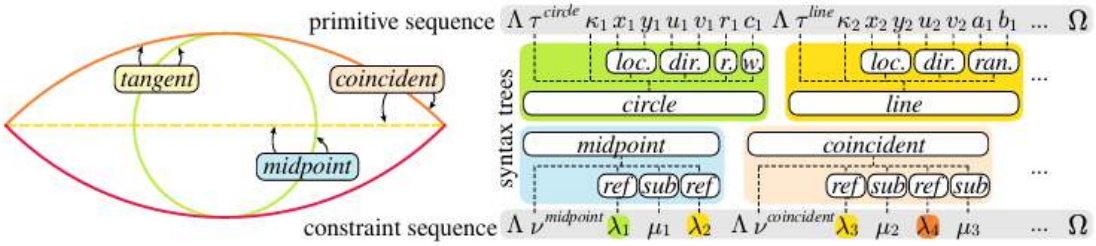


Figura 4.9. Representación de las relaciones en un sketch, entre las geometrías primitivas y las restricciones geométricas. Recuperado de [5].

Como podemos ver en la figura 4.9, se muestra las relaciones que existen en un sketch, entre las geometrías primitivas (por ejemplo una línea, arco, o círculo), y las restricciones geométricas (por ejemplo, tangente, paralelo, punto medio, coincidente). En la figura se observan dos arcos, un círculo, y una línea recta la cual está punteada lo cual representa que es una geometría constructiva. También se observan las restricciones geométricas, que en este caso son 3, tangencia, coincidencia, y punto medio. Las relaciones entre geométricas y restricciones son: el círculo y la línea se unen en su punto medio o centro según se vea; el círculo y los arcos son tangentes entre sí; la línea recta y los arcos son coincidentes en sus extremos o puntos finales según se vea.

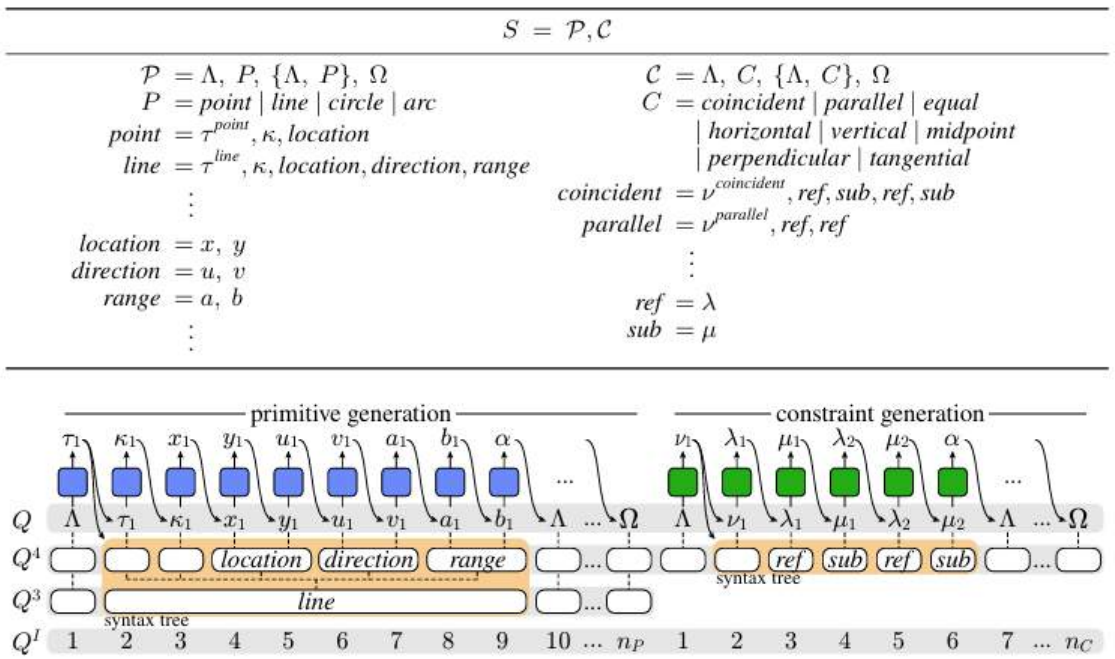


Figura 4.10. Gramática del lenguaje propuesto para describir el sketch con texto. Recuperado de [5].

En la figura 4.10, vemos como se propone la gramática para describir los sketch como texto, en ella se describe el skech como "S" donde tiene un conjunto de primitivos "P" y un conjunto de restricciones "C", ahora estos primitivas puede ser por ejemplo: líneas, puntos, círculos, arcos. Las restricciones geométricas pueden ser: coincidente, paralelo, igual, horizontal, vertical, punto medio, perpendicular, tangente. A su vez cada uno de las geometrías primitivas tiene parámetros asociados, por ejemplo el punto tiene coordenadas, la línea tiene puntos de inicio punto final, longitud entre otras. Luego las restricciones geométricas también tienen parámetros asociados por ejemplo referencias a geometrías, sub referencias, entre otras, según la restricción geométrica en cuestión.

En la tabla 4.2 podemos ver los parámetros de las geometrías primitivas expuestos con mayor detalle, por ejemplo en el caso del arco lo define como las coordenadas de inicio y final, así como del centro.

Tabla 4.2 Parámetros de las geometrías primitivas. Recuperado de [5].

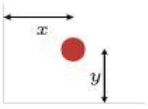
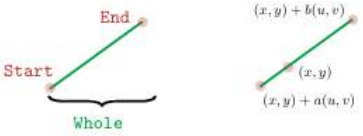
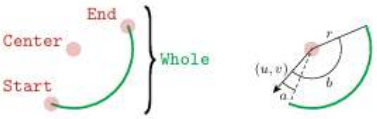
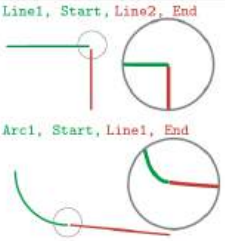
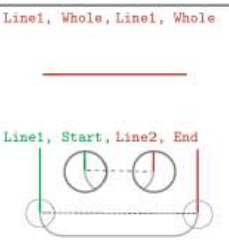
Primitive	Sub-references	Parameters	Visualization
point	`Whole`	x, y	
line	`Start` `End` `Whole`	x, y, u, v, a, b	
arc	`Start` `End` `Center` `Whole`	x, y, u, v, r, c, a, b	
circle	`Center` `Whole`	x, y, u, v, r, c	

Tabla 4.3. Parámetros de las restricciones geométricas. Recuperado de [5].

Constraint	Parameters	Description	Visualization
coincident	$\lambda_1, \mu_1, \lambda_2, \mu_2$	Makes two points coincident. The points can be sub-references of primitives.	
horizontal	$\lambda_1, \mu_1, \lambda_2, \mu_2$	Imposes a horizontal constraint on one/two primitives.	

Ahora en la tabla 4.3 se muestra las parámetros de las restricciones geométricas donde por ejemplo para el caso de la restricción de coincidencia vemos que se usan 4

parámetros, lambda 1 y lambda 2 que serán las geometrías de referencias, y mu1 y mu2, que son los puntos de las geometrías a unir.

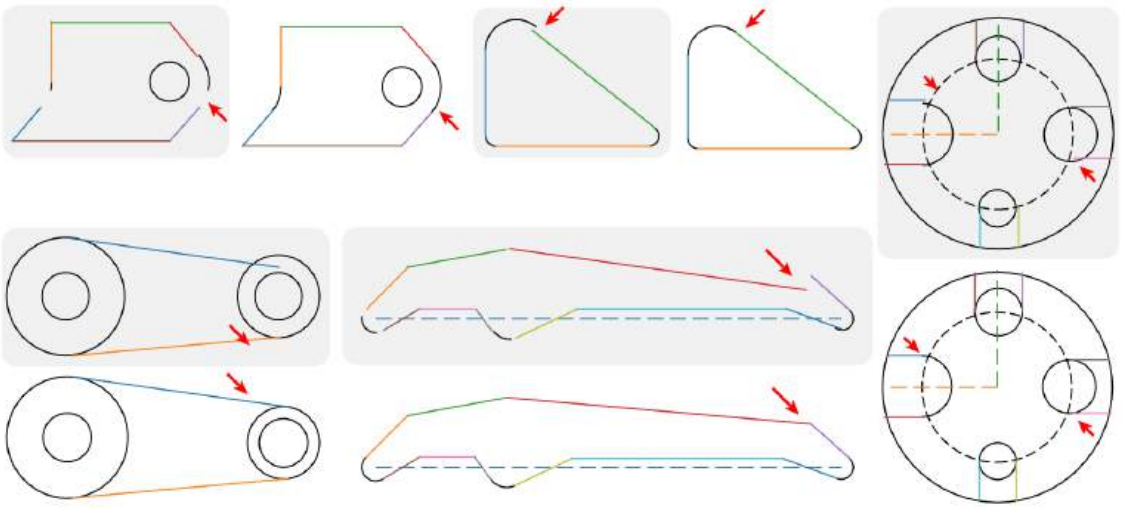


Figura 4.11. Ejemplo de sketch generados de forma automática. Recuperado de [5].

En la figura 4.11 vemos ejemplos de sketch generados de forma automática. Los que tienen fondo gris son sketch antes de la optimización y los que no no tienen fondo son después de la optimización.

Se puede observar que estos sketch son solo representaciones y no se puede hacer alguna modificación con ellos ni mucho menos realizar operaciones volumétricas. Es por ello que el uso o aplicaciones de este trabajo está limitado.

Ahora pasamos a un trabajo que se enfoca en lo archivos CAD y lo sketches es el caso de [6], este trabajo busca generar de forma automática un archivo CAD, recordemos que en los archivos CAD hay una gran variedad de operaciones y aplicaciones por lo que los archivos CAD son complejos, pero como ya se ha comentado, utilizando únicamente geometrías primitivas simples en el sketch, y aplicando únicamente la operación volumétrica de extrusión se pueden generar una gran variedad de geometrías.

En este trabajo se generaron 178,238 modelos CAD, que están representado en forma de " secuencias de comandos CAD", la propuesta se basa en intentar describir el archivo

CAD como texto para de esta forma aprovechar los avances en procesamiento del lenguaje natural, y por así generar "oraciones" que sean "secuencias de comandos CAD".

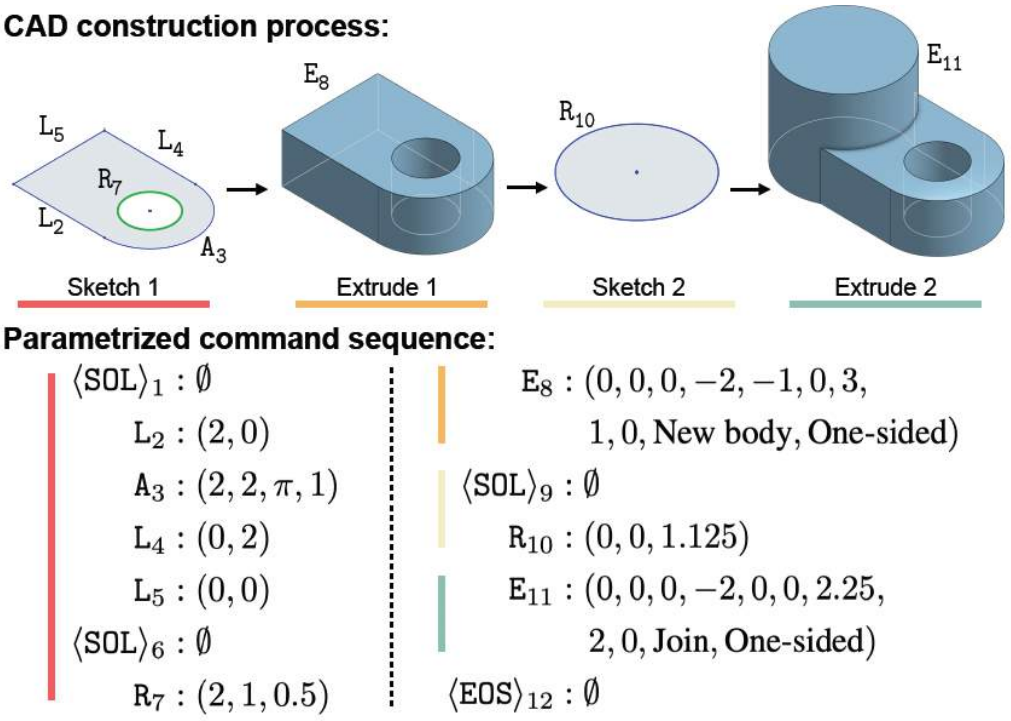


Figura 4.12. Modelo CAD y elementos de la "secuencia". Recuperado de [6].

En la figura 4.12 podemos ver una representación de un modelo 3D, como se mencionó, puede ser un secuencia de comandos CAD, primero un sketch, y sus geometrías primitivas, en este caso inicia con el comando SOL (inicio de bucle), para continuar con las geometrías primitivas, que son línea 2, arco 3, linea 4, y por último linea 5. Luego vemos que se inicia otro bucle siendo este el comando 6, seguido por el comando circulo 7. Después de estos comandos que describen al sketch, o en otras palabras con el sketch finalizado, lo que sigue en la secuencia es una extrusión la cual es la operación 8, la cual genera un "volumen" que se muestra en azul en la parte superior.

Tabla 4.4. Parámetros de los Comando usados. SOL es inicio de bucle, y EOS es final de secuencia. Recuperado de [6].

Commands	Parameters
⟨SOL⟩	∅
L (Line)	x, y : line end-point
A (Arc)	x, y : arc end-point α : sweep angle f : counter-clockwise flag
R (Circle)	x, y : center r : radius
E (Extrude)	θ, ϕ, γ : sketch plane orientation p_x, p_y, p_z : sketch plane origin s : scale of associated sketch profile e_1, e_2 : extrude distances toward both sides b : boolean type, u : extrude type
⟨EOS⟩	∅

Esta combinación de sketch y extrusión, pueden generar una gran variedad de geometría, con una amplia gama en su complejidad según el tipo de sketch, y la cantidad de "pares" de sketch y extrusión. Cabe recalcar que estas son limitadas pero suficiente para describir muchas geometrías.

En la tabla 4.4, se muestran los parámetros de los "comandos" de la "secuencia". Por ejemplo para el caso de los comandos de SOL (inicio de bucle), y de EOS (fin de secuencia), no se requiere un parámetro asociado, ya que estos solo son marcadores de "eventos" en la secuencia.

$$\boldsymbol{p}_i = [x, y, \alpha, f, r, \theta, \phi, \gamma, p_x, p_y, p_z, s, e_1, e_2, b, u]). \quad (1)$$

Ahora para el caso de una línea, solo se requieren las coordenadas del final de la línea, el inicio de la línea debe estar delimitado por la última geometría. Es decir la primera geometría inicia donde termina la última geometría.

En la ecuación 1 vemos los parámetros usados en la generación de las "comandos CAD", son una recopilación de los 16 posibles parámetros que los diferentes comandos tienen. En el caso que ese parámetro no se use se coloca un -1. El tamaño de la "secuencia" se acota a 60 "comandos". Para tener un tamaño constante de la secuencia, en el caso que la secuencia tenga menos de 60 comandos, se realiza un "padding" con el comando "EOS" (fin de secuencia).

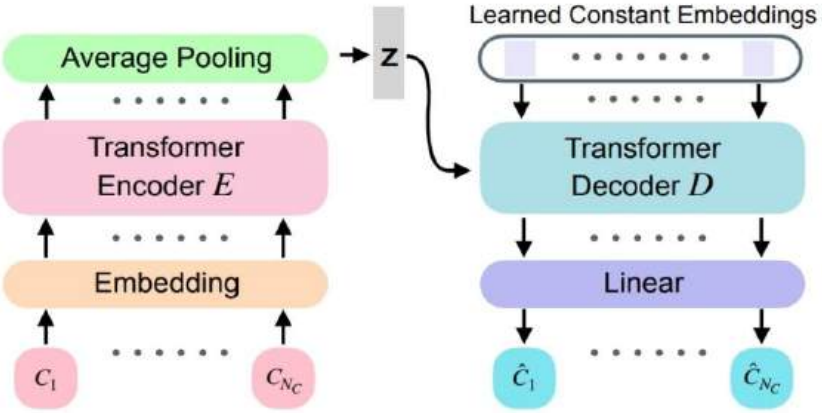


Figura 4.13. Estructura del Modelo de IA. Recuperado de [6].

En la figura 4.13 se muestra la estructura del modelo usado para generar la secuencia de comandos, los datos de entrada van de C_1 a C_{nc} y los datos de salida van de $\hat{C}_1$ a $\hat{C}_{nc}$. Los datos de entrada pasan por una capa de embedding luego pasan al transformer encoder, de ahí a una capa de pooling, y luego un espacio latente "z", de aquí pasan al transformer decoder y luego a una capa lineal para generar la "secuencia de comandos".

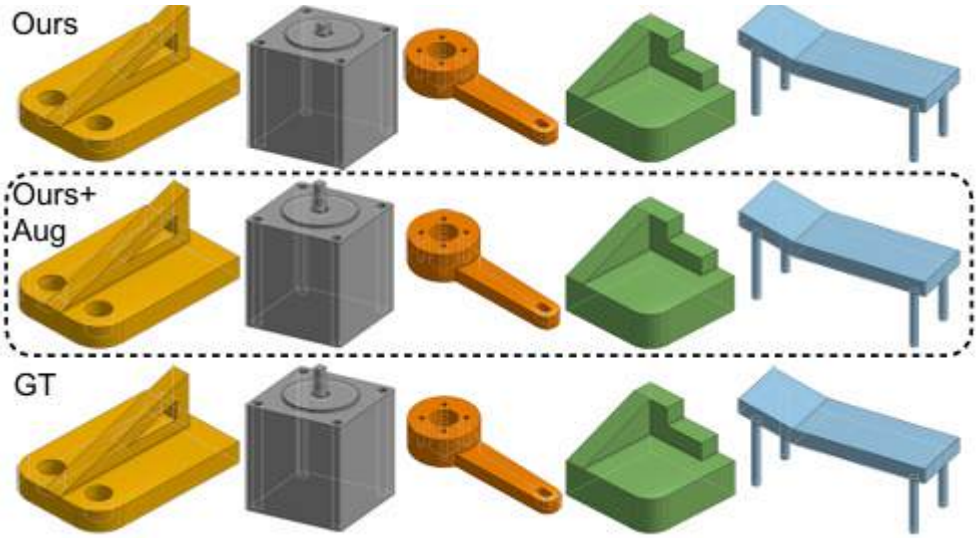


Figura 4.14. Modelos Generados con DeepCAD. Recuperado de [6].

Ahora en la figura 4.14 podemos observar modelos o archivos 3D generados con el modelo propuesto. Se puede observar que la generación es muy adecuada ya que los modelos se parecen mucho a la verdad fundante. Podemos destacar de este trabajo que aborda la generación de archivos 3D, el detalle es que no se enfoca tanto en la parte del CAD, y se enfoca más en la parte generativa. Lo cual resulta interesante pero dista del objetivo planteado en el presente trabajo el cual es reconstruir de forma editable o accesible el sketch para a su vez poder agregar tanto restricciones geométricas, como restricciones dimensionales. Estas modificaciones son elementales para poder usar el archivo CAD en todas sus aplicaciones.

Siguiendo con trabajos que se enfocan en el sketch tenemos el caso de [7] que tomando diseños subidos por la comunidad a la plataforma de "Autodesk Online Gallery", se busca tomar los archivos que usen exclusivamente las operaciones de sketch y extrusión.

La razón de solo usar sketch y extrusión es que estas operaciones representan el 79% de los diseños disponibles en la galería, el objetivo principal es mediante un lenguaje de dominio específico (DSL por sus siglas en inglés), generar las operaciones o secuencias de comando que puedan representar este objeto 3D, y con esto generar una base de datos que pueda ser usada en entrenamiento de modelos de inteligencia artificial.

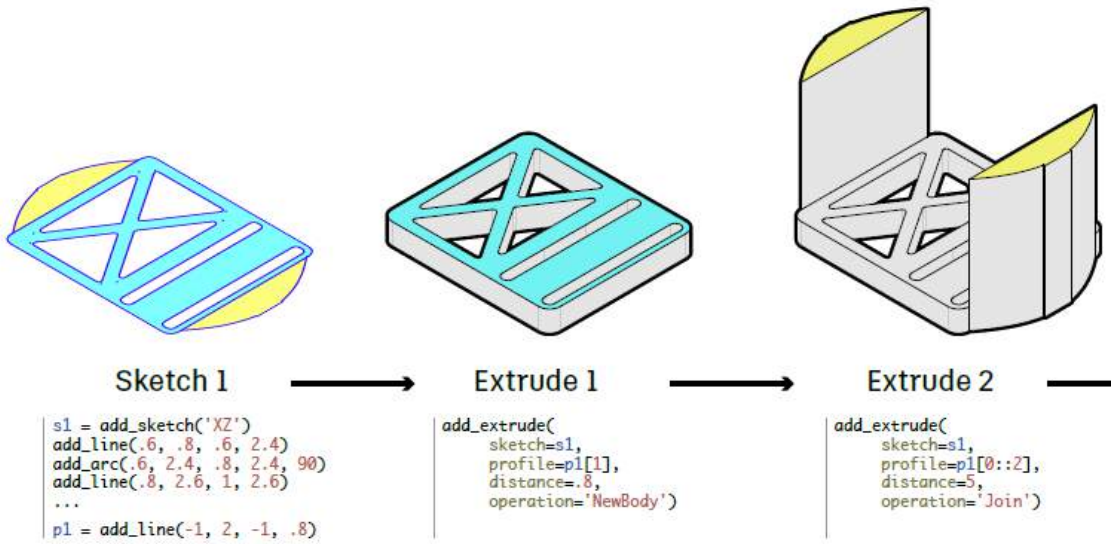


Figura 4.15. Operaciones de Sketch y extrusión. Recuperado de [7].

Podemos ver en la figura 4.15 como la combinación de las operaciones de sketch y extrusión se puede generar geometrías muy complejas. Se observa que por ejemplo se pueden ejecutar dos diferentes operaciones de extrusión para generar dos volúmenes diferentes.

Desde un mismo plano se extuyen dos superficie diferente a difetenes alturas, la primera extrusión parte del area denotada en azul, mientras que la segunda extrusion parte del area denotada en amarillo.

Como podemos ver se genera una geometría compleja con solo 3 operaciones, 1 sketch con dos áreas definidas, y dos operaciones de extrusión.

Cabe aclarar que si bien esta operación es posible y es una práctica habitual en el diseño 3D, por lo general se recomienda que cada operación de extrusión esté asociada a un solo sketch, es decir secuencias de operaciones sketch-extrusión.

La intención de esta restricción es que cada operación volumétrica está asociada solamente a un sketch, ya que en caso de modificar el sketch solo 1 operación se vería afectada.

$$
\begin{array}{lcl}
P & := & G;[X] \\
X & := & S \mid E \\
S & := & \text{add\_sketch}(I);[D] \\
D & := & L \mid A \mid C \\
L & := & \text{add\_line}(N,N,N,N) \\
A & := & \text{add\_arc}(N,N,N,N,N) \\
C & := & \text{add\_circle}(N,N,N) \\
E & := & \text{add\_extrude}([I],N,O) \\
I & := & \text{identifier} \\
N & := & \text{number} \\
O & := & \text{new body} \mid \text{join} \mid \text{cut} \mid \text{intersect}
\end{array}
$$

Figura 4.16. Gramática del lenguaje de dominio específico propuesto para la reconstrucción. Recuperado de [7].

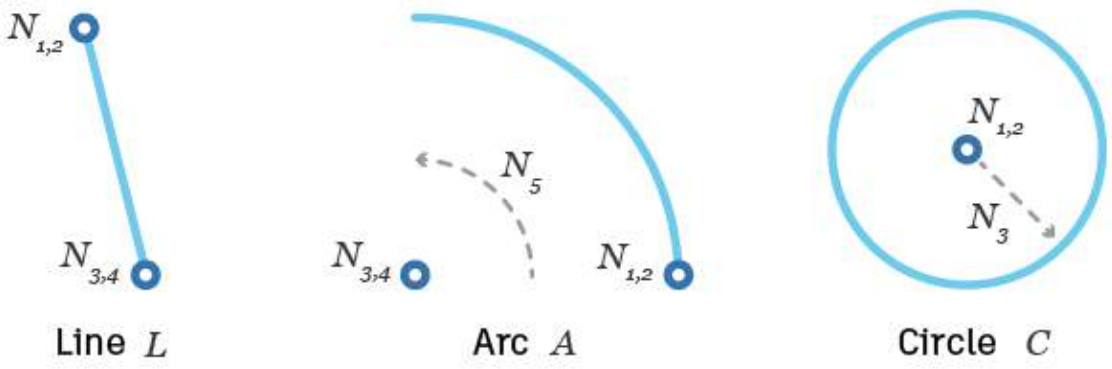


Figura 4.17. Parámetros de las geometrías primitivas de forma ilustrada. Recuperado de [7].

En la figura 4.16 podemos ver la "gramática" usada en el "lenguaje de dominio específico" con sus parámetros, tenemos que P es un "programa", que tiene una geometría G, dada unas secuencia de operaciones X. Las operaciones X pueden ser S (sketch), y E (extrusión). Por su lado el sketch tiene un identificador I, y una serie de comandos D. La serie de comando D pueden ser: Líneas L, Arcos A, y Círculos C. Por

su parte la operación de extrusión E, tiene un identificador I, un parámetro de distancia N en la dirección de la normal, y por último un tipo de operación boolean O.

En la figura 4.17, vemos los parámetros usados en las geometrías primitivas de forma ilustrada. Para el caso de las líneas, tenemos 4 números que serán las coordenadas en x, y del punto de inicio, y luego las del punto final. En el caso del arco tenemos 5 valores los cuales representan las coordenadas de inicio del arco, las coordenadas del centro, y el quinto número es un ángulo desde las coordenadas de inicio en sentido antihorario.

Para el proceso de reconstrucción y análisis de las geometrías que se van generando podemos identificar dos entidades separadas. Por una lado tenemos la geometría actual denotada por G_c y la geometría objetivo denotada por G_t.

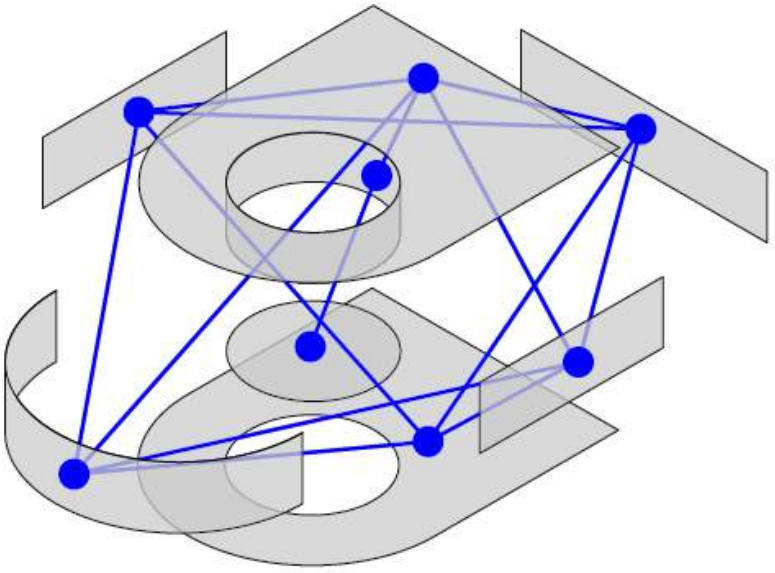


Figura 4.18. Representación de grafos de las caras adyacentes. Recuperado de [7].

En la figura 4.18 podemos ver como una geometría puede ser representada o identificada como grafos, donde los vértices son las "caras" y las aristas son las conexiones entre caras adyacentes. Esto resulta útil en el proceso de reconstrucción de los pasos para generar una geometría objetivo a partir de una geometría actual. Estos ya

que se pueden ir comparando las superficies o caras para ver qué operaciones o geometrías sería necesarias para acercarse a la geometría objetivo.

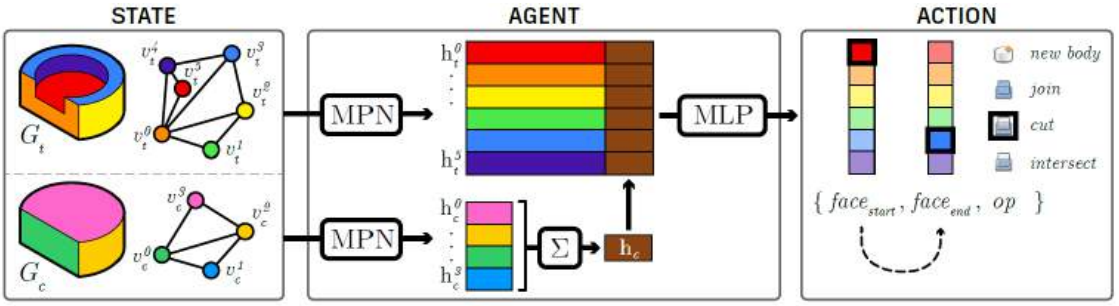


Figura 4.19. Pasos para generar la geometría objetivo desde una geometría actual. Recuperado de [7].

En la figura 4.19 vemos los pasos para generar una geometría objetivo desde una geometría actual. En él vemos como dado un estado con una geometría actual denotada por G_c y la geometría objetivo denotada por G_t. Usando redes de paso de mensajes MPN primero del G_c se generan unos vectores de nodos incrustados, que se suman para generar h_c. A partir de G_t usando otra MPN, se generan otros vectores h_t^0 … h_t^5 que se concatenan con h_c. Usando un perceptrón multicapa se predice la acción a seguir que en este caso es generar un "corte" desde la superficie roja, como "cara de inicio", y hasta la superficie azul o "cara de fin". Con esta acción se acerca la geometría actual a la geometría objetivo.

Se usan diferentes agentes para realizar la reconstrucción de los modelos, por una lado un agente que puede seleccionar de forma aleatoria de todas las acciones posible que será "rand", luego "mlp" que es un mlp simple que no toma ventaja de de redes de paso de mensajes ni de topología de grafos, y por último 3 agentes que si hace uso de grafos, por una lado, "gcn" que es una red de grafos convolucional, luego "gat" que es una red de atención de grafos, y finalmente "gin" que es una red isomórfica de grafos.

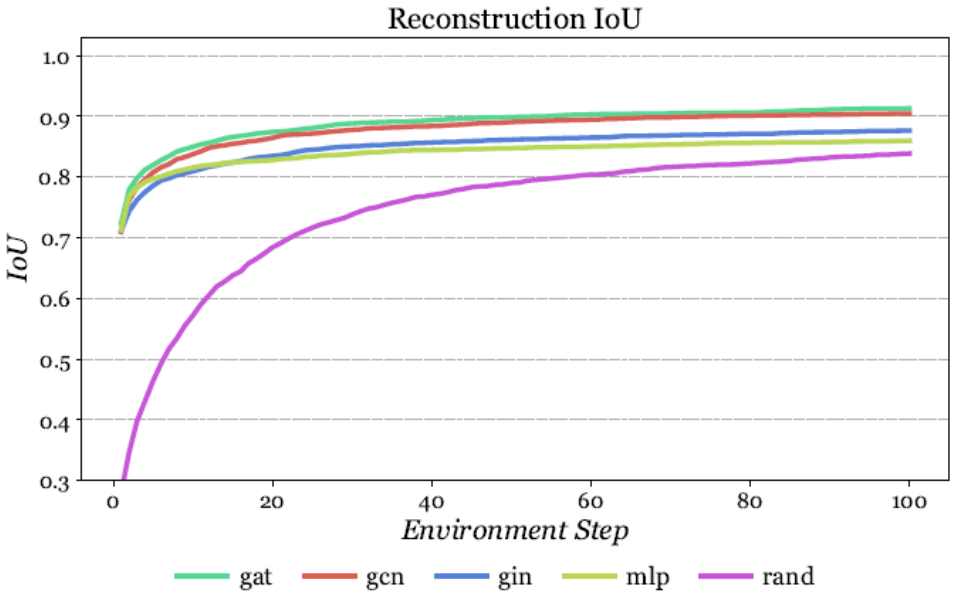


Figura 4.20. Métrica de intersección sobre unión de los modelos a los largo de 100 pasos o acciones. Recuperado de [7].

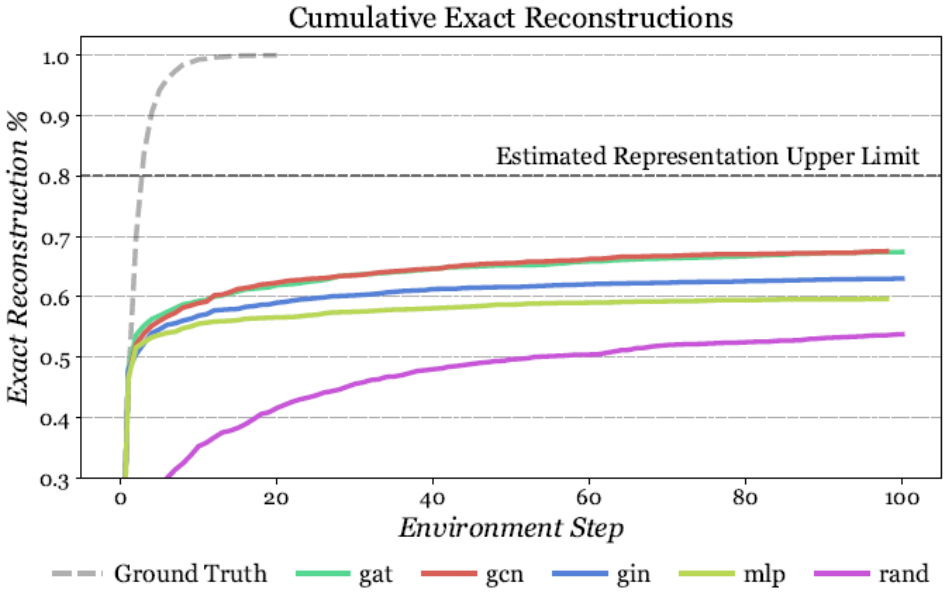


Figura 4.21. Porcentaje de archivos 3D reconstruidos exactamente usando los diferentes modelos, a lo largo de hasta 100 pasos o acciones. Recuperado de [7].

En las figuras anteriores 4.20 y 4.21, podemos ver la métricas de reconstrucción de los modelos, donde se observa que los mejores modelos son “gat” y “gcn” con un IoU de

91% y 90% respectivamente a los 100 pasos. Se observa que el porcentaje de reconstrucción exacta es relativamente bajo ya que de los modelos “gat” y “gcn” solo se pudieron construir exactamente el 67.4% y 67.5% por ciento de los modelos de manera exacta a los 100 pasos o acciones.

También es interesante notar que muchos de los modelos originales toman menos de 20 “pasos” para realizar la geometría mientras que los modelos de generación requieren varias veces más de pasos para poder acercarse a esta reconstrucción.

Figura 4.22. Geometrías reconstruidas. Recuperado de [7].

Ahora en la figura 4.22 vemos algunos ejemplos de los diferentes modelos y datos usando para la reconstrucción. Donde “gt” es la verdad fundante, “aug” es un modelo convolucional entrenado con datos humanos y datos sintéticos, “gcn” es una red de grafos convolucional, “mlp” es un perceptrón multicapa simple, y “rand” es un modelo que genera pasos de forma aleatoria. Vemos que si bien se acercan mucho con métricas relativamente buenas de más del 90% aún tiene diferencias para se exactamente la reconstrucción del diseño CAD original.

Ahora pasamos a un trabajo relativamente similar al trabajo presente ya que busca reconstruir o generar el sketch pero en este caso parte desde dibujos a mano alzada.

La idea parte de que muchas veces en el proceso de diseño para ir plasmando la idea se realizan dibujos a mano, y de cierta forma este dibujo a mano ya es el diseño final, por lo que sería muy útil tener una forma de generar el sketch o dibujo paramétrico partiendo de esos dibujos. En otras palabras poder digitalizar el dibujo a mano, para pasarlo al software CAD.

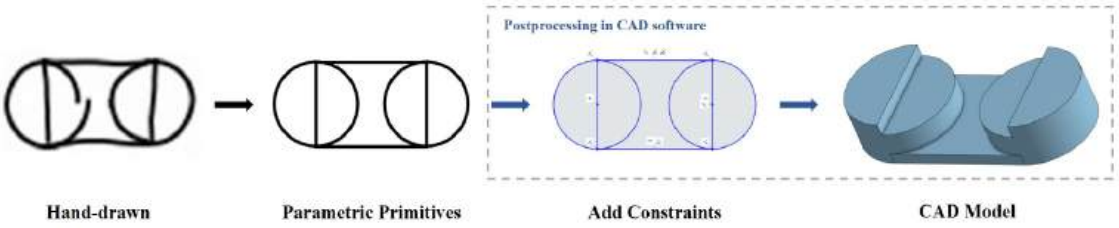


Figura 4.23. Proceso para digitalizar el dibujo a mano alzada. La parte dentro de líneas punteadas es un proceso posterior que no se aborda en este trabajo. Recuperado de [8].

En la figura 4.23. vemos el proceso propuesto para digitalizar el dibujo a mano, también en la parte que está dentro de las líneas punteadas no se abordan en el trabajo sino que se propone como un proceso posterior. Ya que este trabajo solo se enfoca en encontrar o reconocer las geometrías primitivas.

Para identificar las "geometrías primitivas" se definen 4 geometrías básicas, las cuales son línea, círculo, arco, y punto.

Tabla 4.5. Parámetros de las geometrías primitivas. Recuperado de [8].

Line	x_1	y_1	x_2	y_2	0	0
Circle	x	y	r	0	0	0
Arc	x_1	y_1	x_{mid}	y_{mid}	x_2	y_2
Point	x	y	0	0	0	0

En la tabla 4.5 vemos los parámetros que tienen las geometrías primitivas objetivos. Cuando los valores no se utilicen se rellenan con ceros. En el caso de la línea los parámetros son las coordenadas de inicio y de fin. En el caso del círculo, las geometrías son las coordenadas del centro y el radio. En el caso del arco son las coordenadas del punto inicial, punto medio, y punto final. Finalmente en el caso de punto es solo la coordenada del punto.

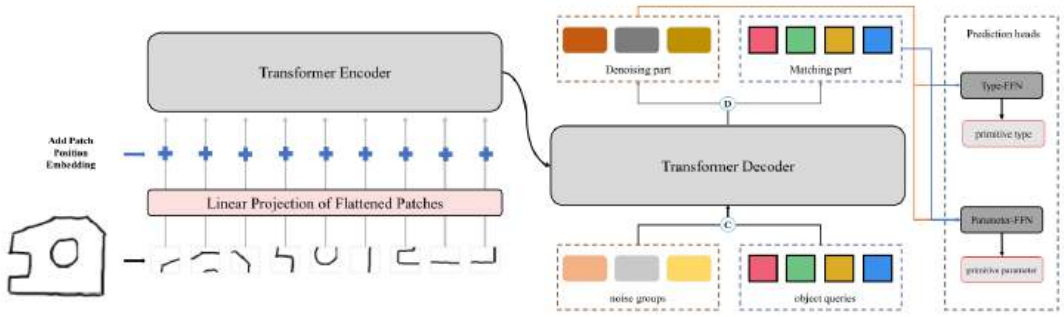


Figura 4.24. Proceso para la identificación y generación del sketch. Recuperado de [8].

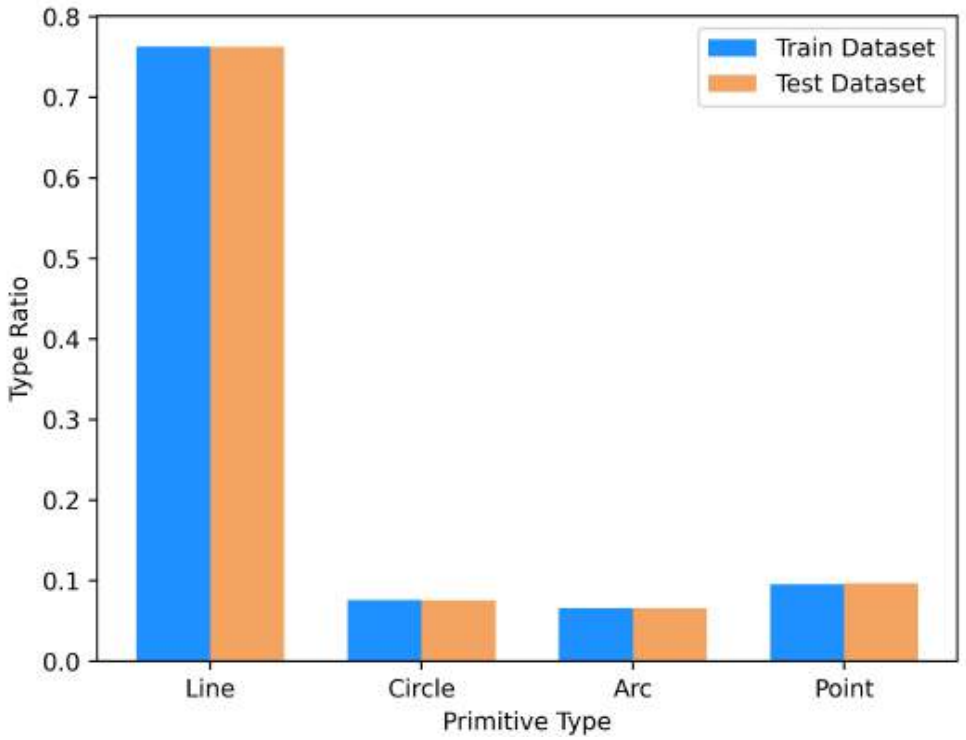


Figura 4.25. Proporción de tipos de geometrías primitivas. Recuperado de [8].

En la figura 4.24 vemos el proceso para identificar y generar el sketch o dibujo paramétrico. Partimos de dibujo a mano alzada. El trabajo consiste en una arquitectura transformer de decoder y encoder. Del lado del encoder, las características de las imágenes son extraídas usando un MLP y se envían a un encoder transformer, con codificación de posición, para obtener las características refinadas de las imágenes. Del lado del decoder, queries de objetos son enviadas al decoder para buscar objetos a través de atención cruzada.

En la figura 4.25, vemos la proporción de tipos de geometrías primitivas que están en el dataset. Se puede observar que las líneas son la mayor cantidad de tipos de geometrías con más del 70% del dataset, mientras que los otros 3 tipos de geometrías tienen aproximadamente el 10% cada una. Por lo que se tiene un desbalance de clases bastante considerable.

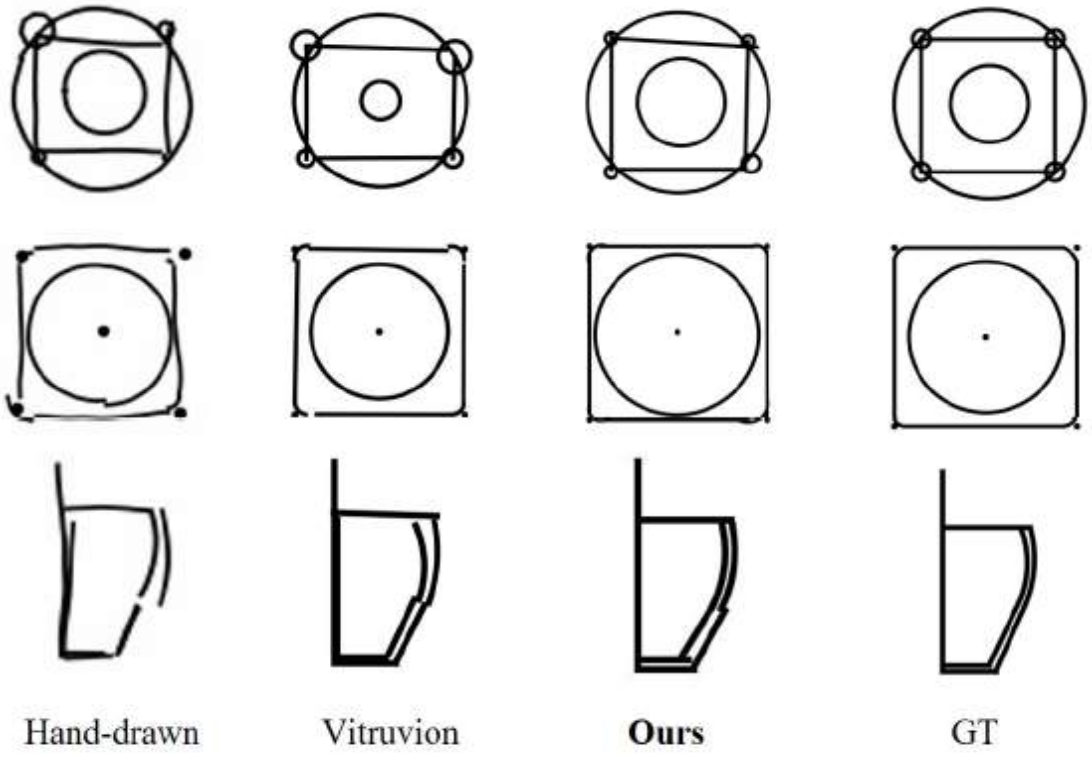


Figura 4.26. Ejemplos de la reconstrucción del sketch. Recuperado de [8].

En la figura 4.26 vemos ejemplo de la reconstrucción donde de izquierda a derecha, se muestra la "entrada" que es el dibujo a mano, luego la comparación con el modelo de otro trabajo llamado vitruvio, seguido del modelo del trabajo [8], y por último la verdad fundante.

Tabla 4.6. Métricas de evaluación comparadas con el modelo de vitruvion. Recuperado de [8].

Metric	Type_acc↑	CD↓	Precision↑	Recall↑
Vitruvion [27]	91.0%	0.0428	66.9%	65.3%
Ours	**93.2%**	**0.0368**	**73.9%**	**71.1%**

En la tabla 4.6 vemos las métricas usadas en la reconstrucción del sketch, la cual se compara con un modelo previo llamado vitruvion. De aquí se puede destacar que en la exactitud de clasificación de geometrías tenemos valores de 93.2% de clasificación.

Podemos mencionar que en este trabajo si bien es interesante el paso de dibujos a mano alzada se considera que no existe mucha información o aplicación para este caso en específico. Además el dataset usado presenta un desbalance de clase muy grande y se enfoca más en la clasificación que en la reconstrucción.

4.4. Reconocimiento de bordes y vértices

Ahora vamos a abordar trabajos que se centren en la identificación de contornos, ya que como se mostrará más adelante nuestro trabajo guarda mucha similitud con la extracción de contornos.

Tenemos el caso del trabajo descrito en [9] dónde a grandes rasgos se busca obtener un polígono del contorno de edificios partiendo de imágenes satelitales.

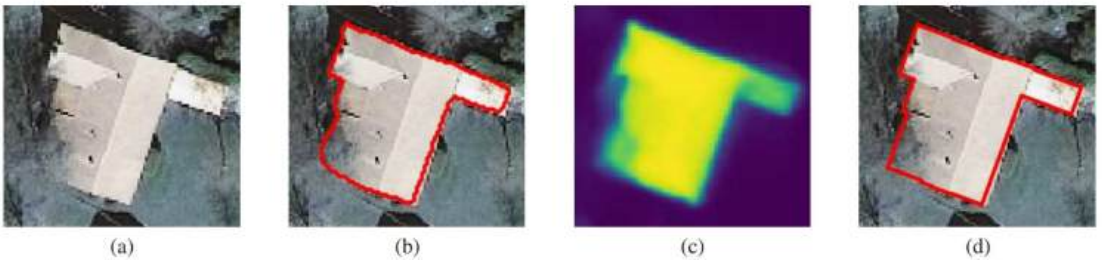


Figura 4.27. Proceso para la identificación del polígono del contorno exterior de un edificio. Recuperado de [9].

En la figura 4.27 vemos de forma simplificada el proceso propuesto para identificar el polígono que describe el contorno exterior de un edificio. Se parte de imágenes satelitales que es el inciso "a". En el inciso "b" vemos la segmentación generada por una red neuronal convolucional. En el inciso "c" vemos el "mapa de probabilidad" el cual también es generado por una Red convolucional. Por último en el inciso "d" tenemos el "polígono" identificado del contorno exterior del edificio.

Se menciona que el mapa de probabilidad toma valores de 0 a 1, para clasificar si un píxel es parte del edificio o no. Está clasificación se realiza utilizando un límite de 0.5. Dónde si la probabilidad de que ese píxel pertenezca al edificio es mayor que 0.5 se asignan la clase 1, es decir que es parte del edificio. Del contrario se asignan la clase 0, es decir que no forma parte del edificio.

En este trabajo se busca analizar los contornos mediante una transformación que convierte al contorno en una "gráfica" "tiempo ángulo".

En la figura 4.28 se ve la gráfica de "tiempo ángulo" generada con la transformada propuesta "RGA" que es por sus siglas en inglés, gradiente relativo del ángulo. Está gráfica describe al contorno partiendo de un punto inicial, como una gráfica que va variando en ángulo relativo a los otros puntos y al "tiempo". Una forma de entender al tiempo en esta gráfica es la longitud del perímetro a partir de un punto inicial.

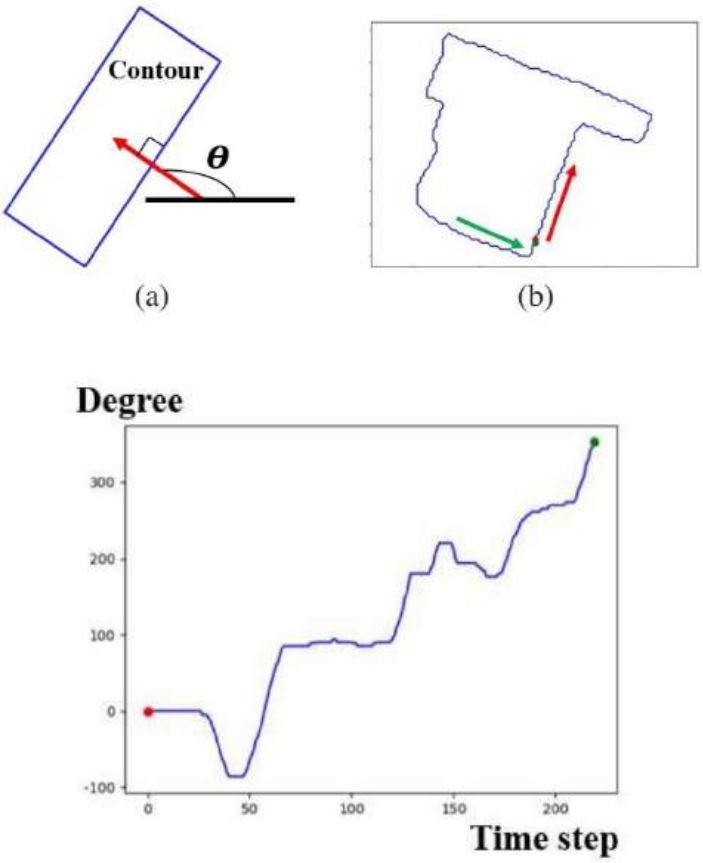


Figura 4.28. Gráfica "tiempo ángulo" del contorno. Recuperado de [9].

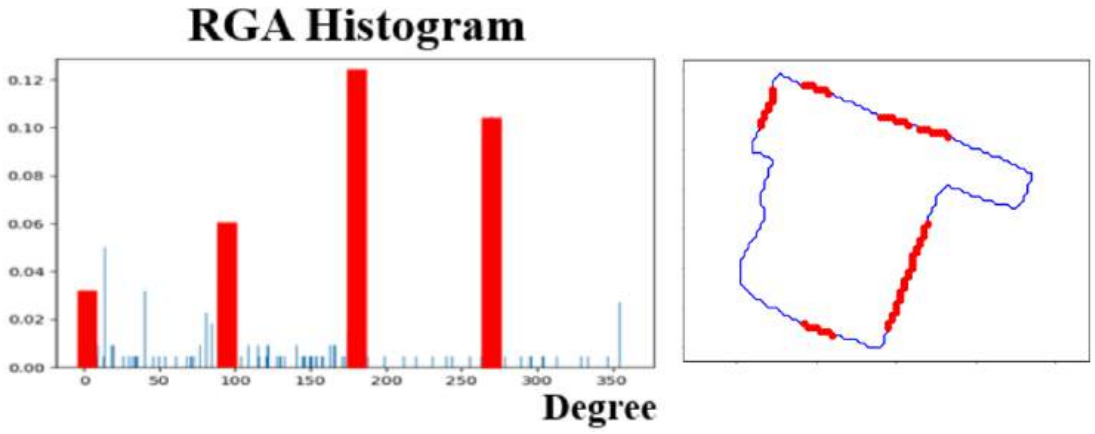


Figura 4.29. Análisis de histograma de los ángulos. Recuperado de [9].

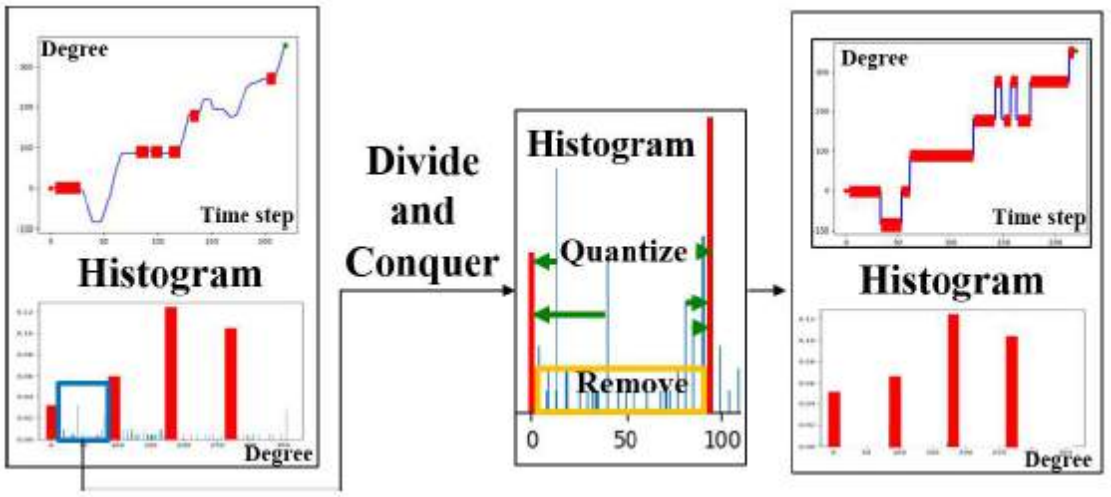


Figura 4.30. Cuantización del histograma. Recuperado de [9].

En la figura 4.29 vemos que los valores convertidos usando el "RGA" se analizan en un histograma donde valores similares de ángulo se agrupan, podemos ver qué estos valores similares también coinciden con segmentos del contorno. Por lo que es un acercamiento en la identificación de los contornos.

Vemos cómo se analizan los elementos que tienen más o menos un ángulo similar. y se puede observar que estos pertenecen a líneas rectas similares, pero algunos están entre cortado o segmentados.

Ahora en la figura 4.30 vemos la cuantización del histograma, donde los "elementos" que tienen ángulos similares se remueven y se agregan a los "elementos" con ángulos similares, de esta forma se "refina" aún más el reconocimientos de las geometrías.

En la Figura 4.31 se muestra el cálculo de los vértices. Ahora con las "líneas" identificadas en su mayoría, lo que se busca analizar los "vértices" esto se realiza al calcular la intersección entre dos líneas como se muestra en la sección "a". Por otro lado en el caso de dos líneas paralelas como se muestra en la sección "b" se realiza el cálculo de los extremos y comparando la información del histograma se agrega una nueva línea como se muestra en azul.

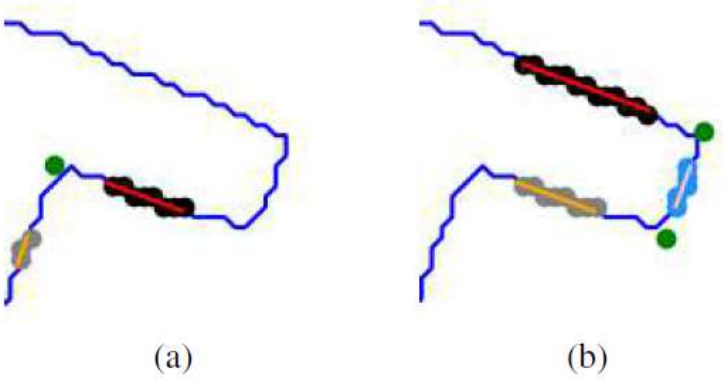


Figura 4.31. Cálculo de los vértices. Recuperado de [9].

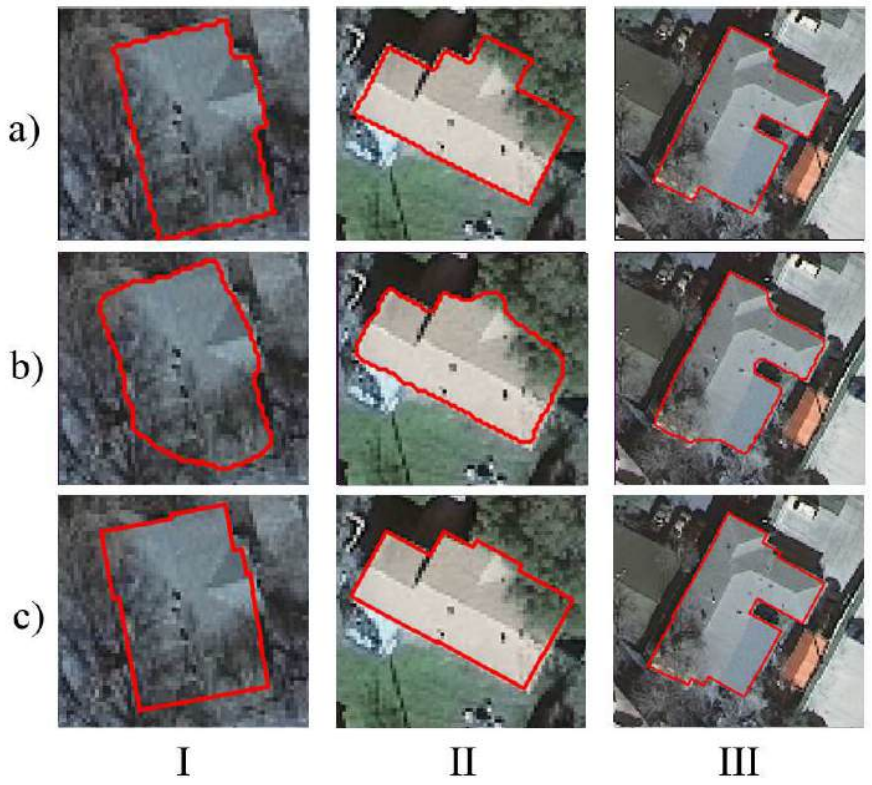


Figura 4.32. Resultado del análisis de contornos de los edificios. Recuperado de [9].

En la figura 4.32 vemos el resultado de la generación del polígono desde imágenes satelitales donde el renglón "a" representa la verdad fundante. El renglón "b", la máscara de segmentación. El renglón "c" es el polígono identificado. Se puede observar

de forma cualitativa que las líneas del contorno en el renglón “c” son más definidas por el tipo de análisis que se realizó.

Si bien los resultados que se reportan con la métrica de IoU son de solo el 78% se observa que cualitativamente mejoran respecto a la verdad fundante.

Ahora pasamos al trabajo descrito en [10] dónde similar al trabajo anterior se busca identificar el contorno de los edificios, pero estos al encontrar los vértices para poder generar un polígono del contorno exterior del edificio.

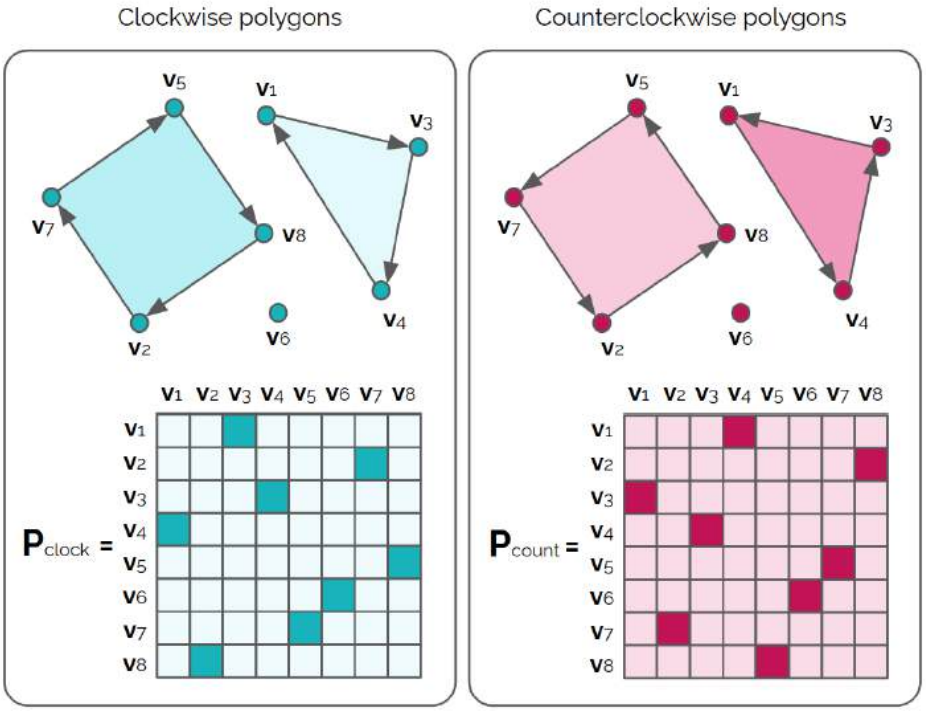


Figura 4.33. Matriz de permutaciones de los vértices. Recuperado de [9].

En la figura 4.33, vemos la matriz de permutación de los vértices. En azul está la matriz en sentido horario, por ejemplo el vértice v1 se conecta en sentido horario con el vértice v3. Por otro lado en rosa tenemos la matriz en sentido antihorario, en este caso el vértice v1 se conecta con el vértice v4. En este trabajo se busca representar los polígonos como una matriz de permutación que describe cómo los vértices están conectados unos con otros. En otras palabras las aristas.

Se considerarán ciertas condiciones para esta matriz de conexiones: La primera cada vértice tiene como máximo una conexión en sentido horario, y otra en sentido antihorario. La segunda, la matriz de permutación en sentido horario es la transpuesta de la matriz en sentido antihorario. Tercera, los vértices que caigan en la diagonal de la matriz pueden ser descartados, otra forma de verlo es que los vértices que no están conectados a nada más que a sí mismo, no forman parte del polígono, por ejemplo el vértices v6.

El modelo de polyworld está compuesto por tres partes. La primera es una red de detección de vértices que está en un set de posibles esquinas de los edificios. La segunda, una red de atención gráfica que agrega información a los vértices y refina su posición. La tercera, una red de conexión óptima que genera las conexiones entre los vértices. Dada una imagen de entrada el modelo genera la posición de las esquinas de los edificios, y una matriz de permutación válida.

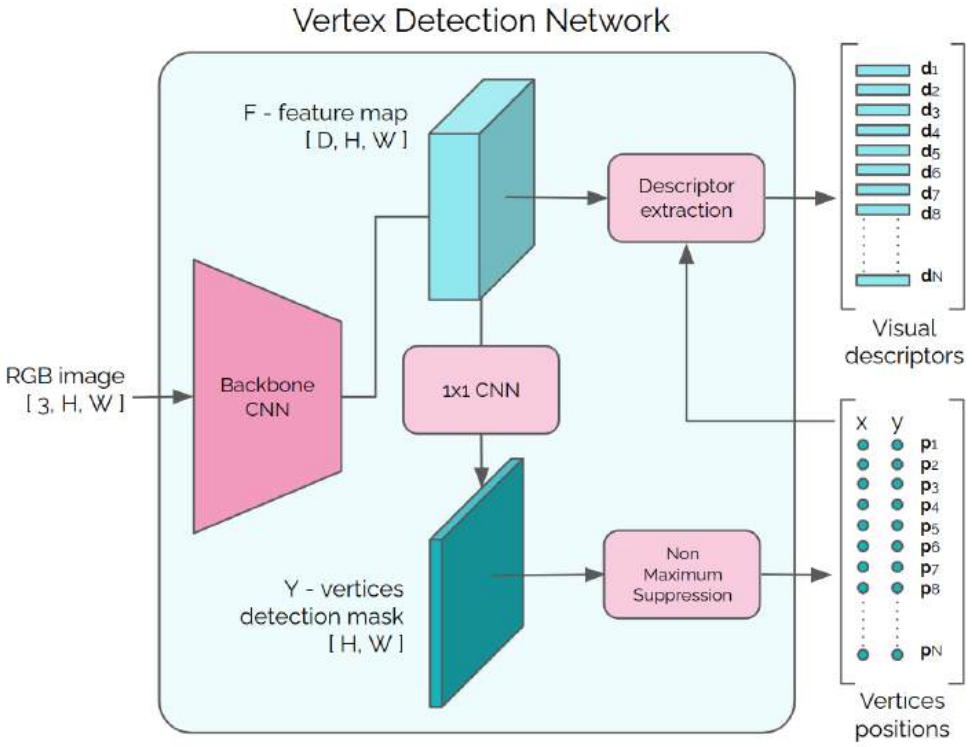


Figura 4.34. Red de detección de vértices. Recuperado de [10].

En la figura 4.34 vemos la red de detección de vértices. En ella vemos que la entrada son las imágenes en 3 canales de colores de las imágenes satelitales que entran a una red convolucional, esto genera un mapa de características, y una mapa de detección de vértices. Luego usando un algoritmo de supresión de no máximos, se remueven vértices no deseados, y se obtienen N posiciones que corresponde a los picos más altos en la máscara de detección. Luego se extraen los descriptores visuales del mapa de características en cada ubicación generada por el algoritmo de supresión de no máximos.

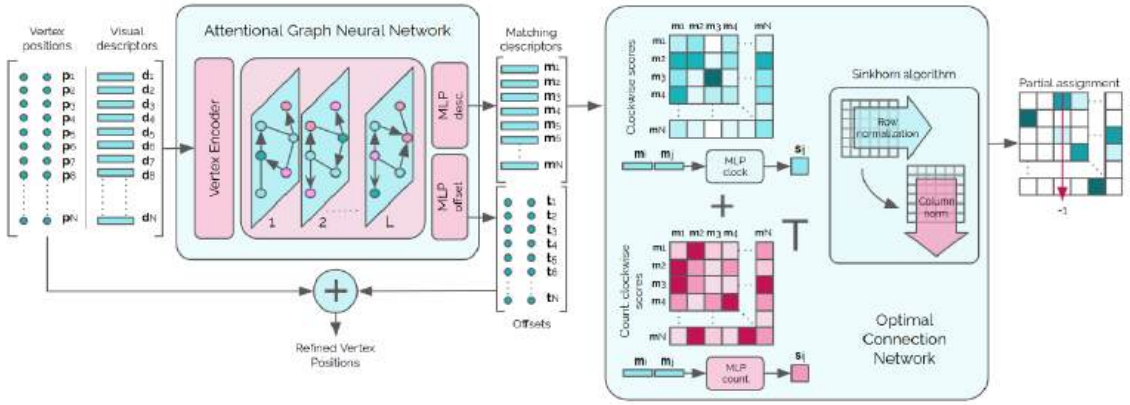


Figura 4.35. Red de atención gráfica y red de conexiones óptimas. Recuperado de [10].

En la figura 4.35 vemos las redes de atención gráfica y la de conexión óptimas. En esta imagen vemos como el "mapa de vértices de posición" junto con los "descriptores visuales" se pasan por un encoder para mapearlos en un solo vector. Se añade una cantidad "L" de capas de atención para incrementar su distintividad.

El modelo regresa unos desfases "t", y sus descriptores correspondientes "m". Los desfases son usados para refinar las posiciones de los vértices, mientras que los descriptores son propagados por la red de "conexiones óptimas" para una matriz de calificación "N x N", lo cual genera las matrices de permutación usando el algoritmo de Sinkhorn.

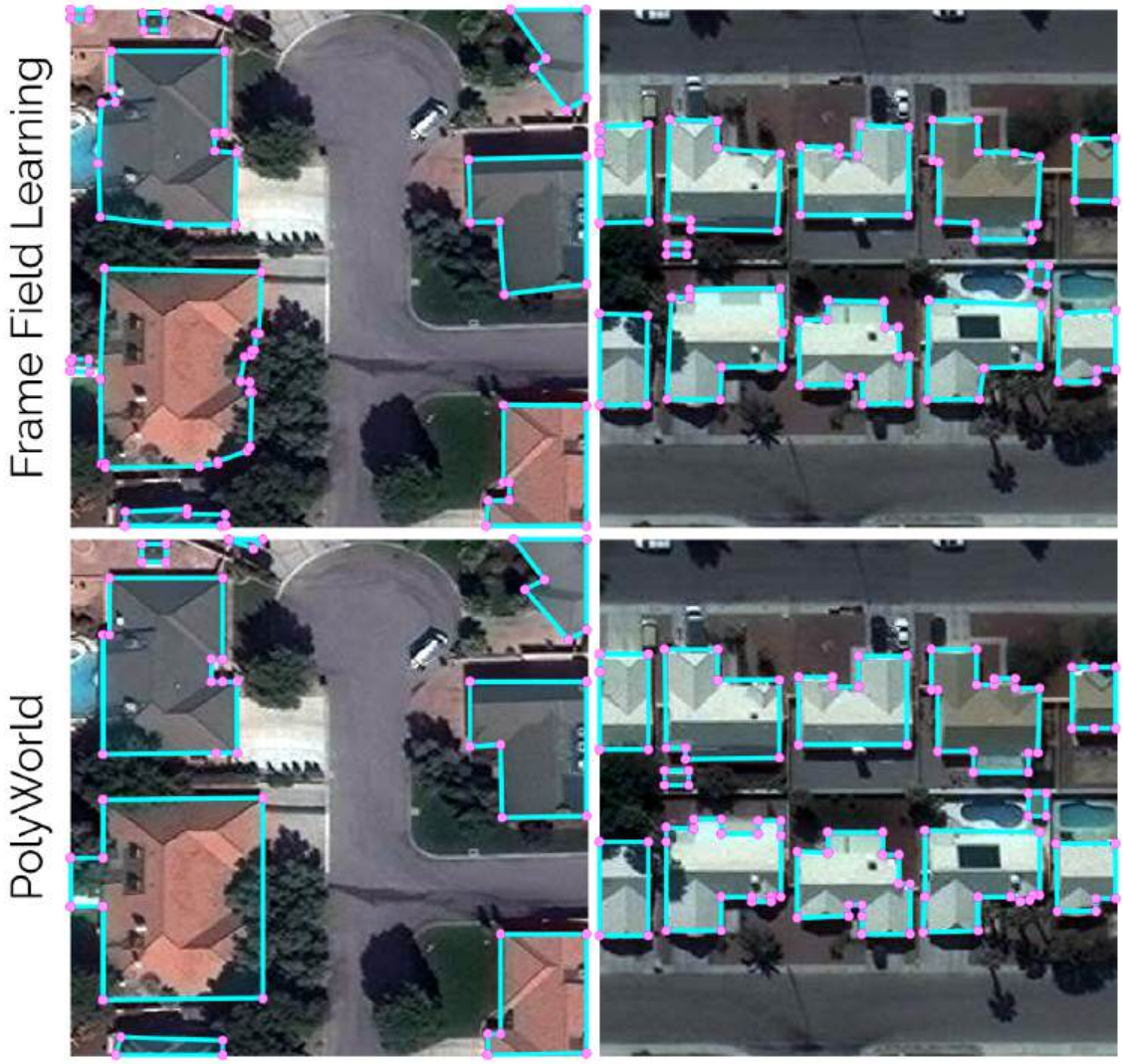


Figura 4.36. Comparación de los resultados de polyworld con "Frame Field Learning". Recuperado de [10].

En la figura 4.36 vemos una comparación entre el método del trabajo polyworld y Frame Fiel learning. Donde se aprecian mejorar en trabajo de poly world. En específico se cuenta con una presión promedio del 63.3 y un recall de 75.4 en comparación con frame field learning de 61.3 y de 64.9 respectivamente. También se tiene métricas de IoU de 91.3 en Polyworld frente a 84.1 de FFL.

Ahora pasamos a un trabajo similar que busca reconstruir el polígono del contorno de edificios en imágenes satelitales. En el trabajo descrito en [11] se busca generar como salida un "campo de marcos" para describir los polígonos, usando una red neuronal profunda, con esto se logra mejorar la calidad de la segmentación.

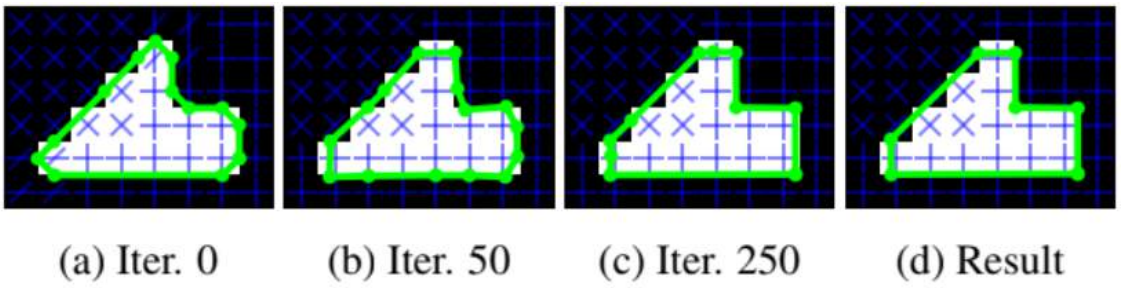


Figura 4.37. Alineación de los polígonos al "campo de marcos" . Recuperado de [11].

En la figura 4.37 vemos como el campo de marcos ayuda a mejorar el polígono resultante. Esto se logra ya que el campo de marcos logra dar mayor referencia a secciones que pueden ser complicadas o difíciles de identificar.

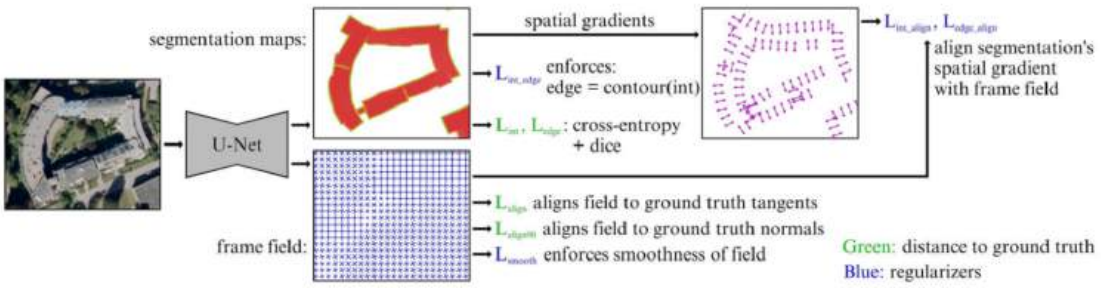


Figura 4.38. Modelo general usado para general el campo de marcos. Recuperado de [11].

En la figura 4.38 vemos a grandes rasgos la estructura del modelo propuesto. Donde en la entrada tenemos imágenes en color de edificios satelitales. Las imágenes pasan por una U-net y tenemos tres salidas por unas las una máscara de bordes, una máscara de región interior. Las dos anteriores salen en la imagen como mapa de segmentación. Por último tenemos el campo de marcos que se ve en azul. También se puede considerar el "mapa de segmentación" como "mapa de clasificación". Los sectores azules son regularizadores que buscan mejorar la "suavidad" de los bordes. y por su lado la parte verde es la pérdida entre la verdad fundante y la predicción.

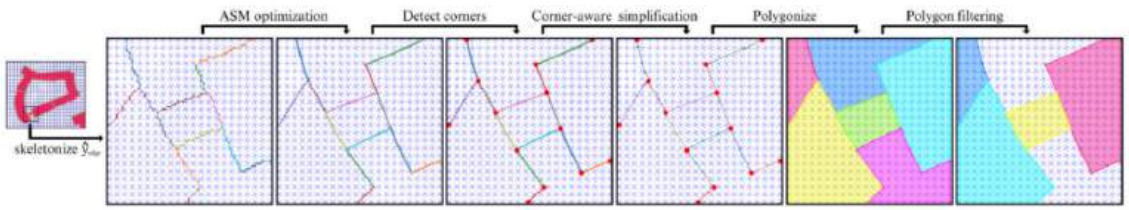


Figura 4.39. Algoritmo de post procesado para generar el polígono final. Recuperado de [11].

En la figura 4.39 vemos el Algoritmo de post procesado para generar el polígono final. Donde de la salida mostrada anteriormente del mapa de clasificación y el campo de marcos se genera un esqueleto de los bordes. Primero para pasar por un proceso de optimización para mejorar los bordes. De aquí pasamos a la detección de esquinas. Una vez con esta información se identifican regiones que puedan ser polígonos. Finalmente se identifican los "polígonos finales" que representan los polígonos de los edificios.

Tabla 4.7. Resultado de IoU y error tangent. Recuperado de [11].

Method	mIoU ↑	Mean *max tangent angle errors* ↓
Eugene Khvedchenya[1], simple poly.	**80.7**%	52.2 °
ICTNet [6], simple poly.	**80.1**%	52.1°
UResNet101 (no field), simple poly.	73.2%	52.0°
Zorzi et al. [41] poly.	74.4%	34.5°
UResNet101 (with field), **our** poly.	74.8%	**28.1**°

En la tabla 4.7 vemos los resultados contra varios métodos. Si bien se observa que el modelo propuesto solo tiene una IoU del 74.8% vemos que mejora el error tangente a 28.1 grados. por lo que representa un avance en la alineación de los bordes que justo se logra con el campo de marcos.

Ahora pasamos al trabajo descrito en [12] donde se enfoca también en la detección de "esquinas" o bordes pero enfocado a aplicaciones de alta velocidad. En este caso tenemos que el modelo se enfoca en videos, por lo que tener el procesamiento en "tiempo real" es muy importante.

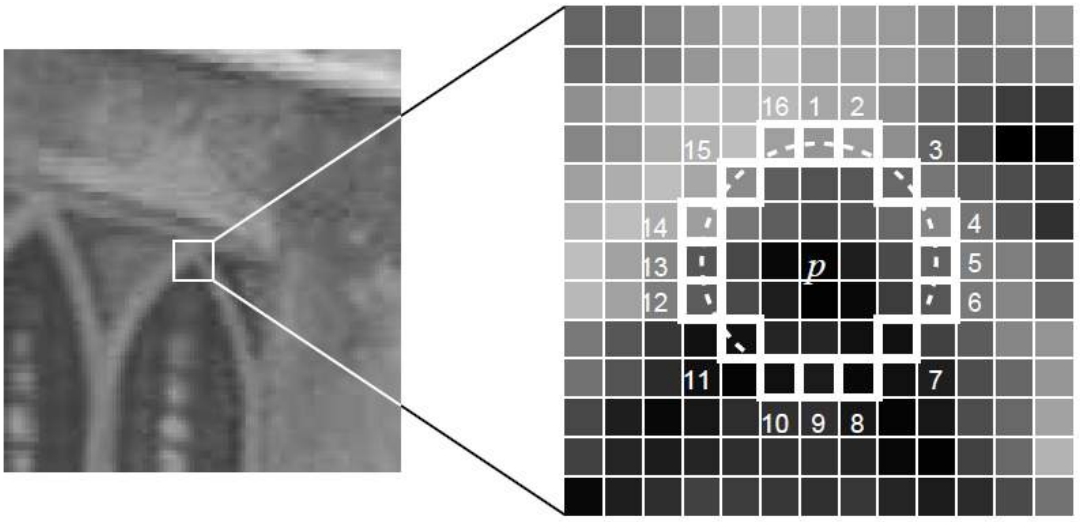


Figura 4.40. Detección de posibles "esquinas". Recuperado de [12].

En la figura 4.40 vemos el criterio para la detección de posibles esquinas. Para considerar un posible "candidato de esquina", se revisan los 16 píxeles que están a su alrededor como se muestra en la imagen. Para revisar el pixel central "p" contra los 16 píxeles que lo rodean se considera una ajuste, "t" el cual sumando al valor del pixel propuesto puede darnos uno de 3 estados, que los píxeles que lo rodean, son "d" más obscuros, "s" similares, o "b" maps brillantes.

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \leq I_p - t \\ s, & I_p - t < I_{p\to x} < I_p + t \\ b, & I_p + t \leq I_{p\to x} \end{cases} \quad (2)$$

Ahora en la ecuación 2 vemos cómo se van clasificando los píxeles según la intensidad de brillo del píxel a analizar "Ip" y el ajuste a la tolerancia. Donde según la comparación, "d" es más oscuro, "s" es similar, "b" es más brillante.

Ahora repitiendo el proceso para todos los píxeles de la imagen se obtienen los mapas para píxeles tanto oscuros, similares, y brillantes. Con este mapa se compara para revisar si son parte de las "esquinas" o no. De esta forma se logra obtener un árbol de decisiones que logra clasificar los píxeles en esquinas o no.

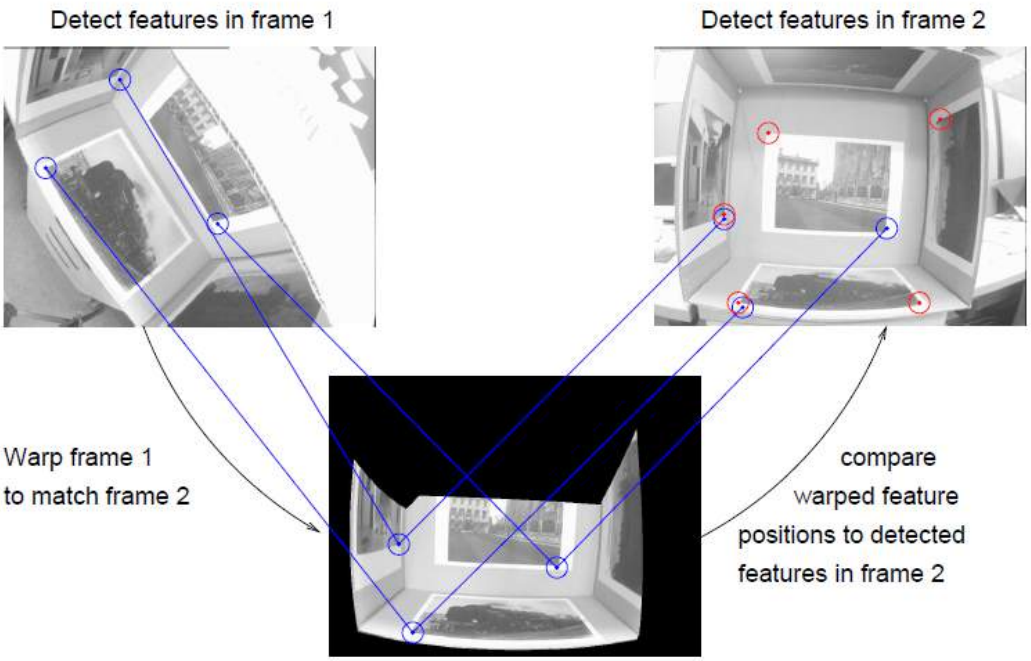


Figura 4.41. Detección de las "esquinas" en diferentes orientaciones. Recuperado de [12].

Ahora en la figura 4.41 vemos cómo las mismas "esquinas" son detectadas en diferentes ángulos y en diferentes "imágenes" o "tomas". En otras palabras vemos cómo se pueden extraer puntos claves en el video. Se puede observar por qué la esquina del marco es identificada por ejemplo en posición vertical, o en posición diagonal de cierta forma tomándola como referencia.

De esta forma a lo largo del video se puede tener consistencia en las "características" o puntos claves identificados. Esto puede a su vez ayudarnos a identificar objetos o realizar la orientación de la toma tomando esos puntos de referencia para realizar el ajuste.

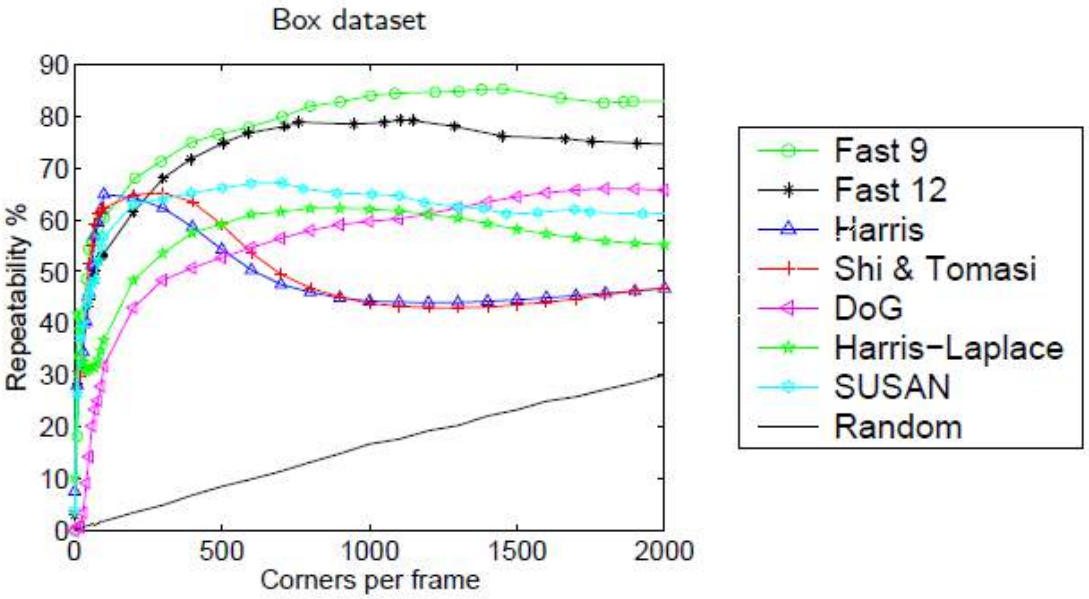


Figura 4.42. Comparación con otros métodos de detección de "esquinas". Recuperado de [12].

En la figura 4.42 podemos ver la comparación de los métodos del trabajo tanto en sus versiones 9 como 12 con otros métodos de detección de esquinas. Se puede observar que en ambos casos hay mejor detección de estas esquinas siendo de los dos anteriores la mejor versión 9.

Como se muestra en los antecedentes existen varios trabajos que abordan el análisis de objetivos 3D, es un campo de estudio con versos objetivos y puntos de partida.. De forma general el objetivo de este proyecto es reconstruir el dibujo paramétrico desde imágenes y archivos STL. Se observa que los trabajos analizan archivos STL no llegan a recrear el dibujo paramétrico. Por otro lado los trabajos que inician con imágenes hacen uso de imágenes satelitales, y se enfocan en contornos de edificios, y no tanto en el dibujo paramétrico. Por su parte los trabajos que trabajan o analizan el sketch o archivos 3D no abordan realmente el problema de la "reconstrucción", ya que se enfocan en analizar o hacer ligeras modificaciones pero no en reconstruir los archivos 3D, además ninguno de ellos usa una representación estándar del sketch, más bien usan representaciones del sketch ya sean propietarias, o propias, propuestas en el trabajo

presentado, lo cual dificulta su análisis y reproducibilidad. Las propuestas anteriores parten de puntos de inicio diferentes al propuesto en este trabajo, o no llegan a generar el sketch o dibujo paramétrico en la interfaz de usuario gráfica lo que limita las aplicaciones y modificación de geometrías.

5. Marco Teórico

En el presente trabajo se utilizaron redes neuronales convolucionales, variando las arquitecturas por eso se describen varios de los componentes y operaciones relacionadas con este tipo de redes. Además para el análisis de los archivos STL, se realizaron cálculos usando algunos conceptos de álgebra lineal que se describen a continuación.

Se agrega también un pequeño apartado donde se muestran algunas operaciones y técnicas propias del diseño mecánico en específico usando el software de diseño mecánico. Las cuales son operaciones que se le pueden realizar al "producto" o resultado del presente trabajo el cual es el sketch o dibujo paramétrico.

5.1. Convoluciones

Las redes neuronales convolucionales son un tipo de red enfocada en procesar datos que tiene forma de tabla, se pueden entender como tablas de 1 renglón o columna que sería de 1 dimensión o como tablas de 2 dimensiones por ejemplo imágenes, o en otras palabras una tabla de píxeles. Es por esto que las convoluciones son buenas para procesar imágenes. La convolución es un tipo de operación matemática como la suma o la resta. Es un tipo de operación lineal dado que mantiene los principios de superposición(aditividad), y homogeneidad(escalado). Esto por ejemplo quiere decir que sumar dos señales y luego realizar la convolución es igual a sumar la convolución individual de cada una de las señales.

En una forma muy general la convolución es una operación matemática que combina dos funciones. El resultado es una salida donde la primera función es modificada por la segunda función. La segunda función es una versión invertida de sí misma y que se va desplazando sobre la primera función. En específico para valores discretos es la "suma"

que mide la superposición de dos señales a medida que una se desplaza sobre la otra. Para caso de valores continuos sería la integral. Ya que las imágenes son valores discretos nos quedamos con la definición de la suma.

Como se mencionó para valores discretos la convolución es una sumatoria. Lo cual también se puede entender como una suma de valores con diferentes pesos que afecta a una función. Los diferentes pesos o "importancia" dependen de la segunda función la cual se invierte y se recorre sobre la primera función.

Una forma de entender esto podría ser que se quiere medir la posición de un objeto que se mueve que puede estar dada por la función x(t). Si se considera que el sensor es susceptible a perturbaciones como ruido, lo ideal será realizar un promedio de las mediciones. En este caso las medición más reciente tienen mayor validez o importancia, en otras palabras mayor "peso". Para asignar esa importancia se puede usar una función de pesos w(a) donde a es la edad de la señal.

$$s(t) = \int x(a)w(t - a)da \tag{3}$$

En la terminología de redes convolucionales se suele llamar a la función x(t) como la entrada. La función de "pesos" se le conoce como "kernel", y a la salida s(t) como "mapa de características" [13]. Esto se describe en la ecuación 3.

En la ecuación 4 se muestra la fórmula usada en el contexto de redes neuronales convolucionales, que en realidad se llama correlación cruzada ya que no se invierte el kernel, pero esta es una convención ya que se suele llamar como convolución aun cuando el kernel no se invierte.

$$(K * I)(i, j) = \sum_m \sum_n I(i + m, j + n) \cdot K(m, n) \tag{4}$$

donde: K es el kernel; I es la imágen; m y n son las coordenadas del kernel; i,j son las coordenadas del mapa de características resultantes.

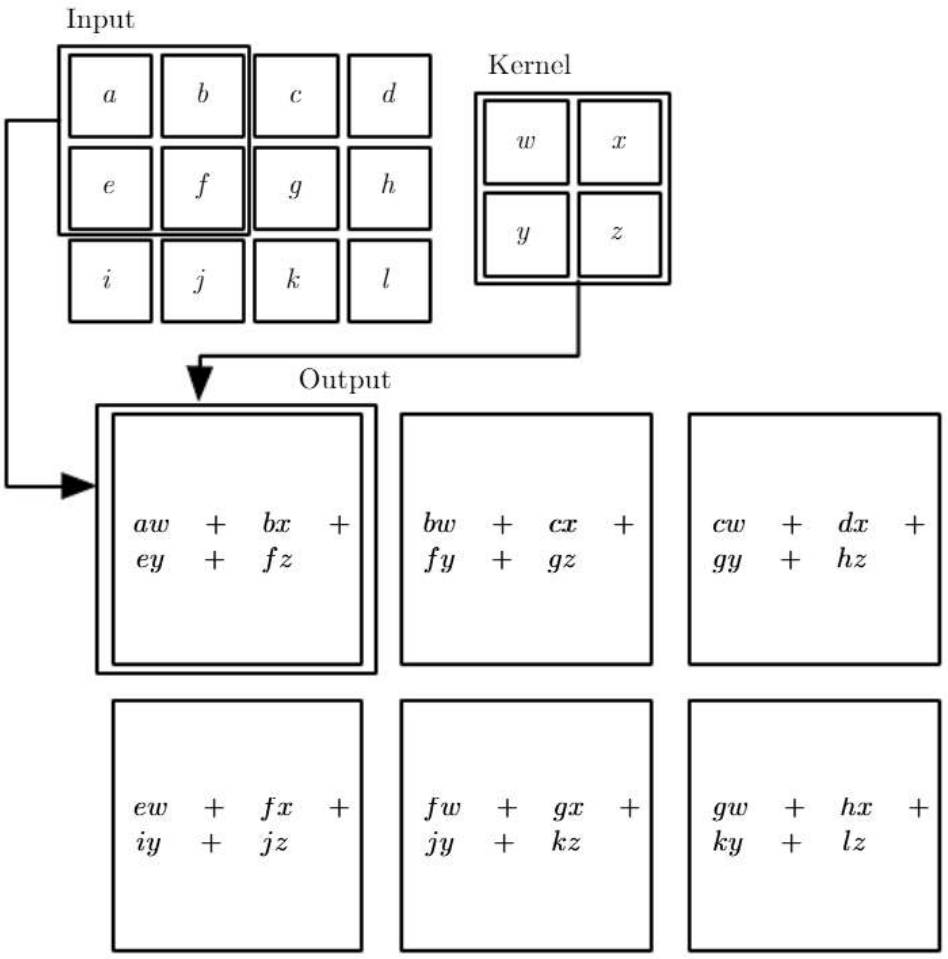


Figura 5.1. Ejemplo sencillo de una convolución de dos matrices. Recuperado de [13].

En la figura 5.1 se muestra un ejemplo simple de una convolución de matrices. En ella se observa la convolución de la entrada "input" con la máscara o "kernel". En este caso la entrada tiene un tamaño de 3 renglones por 4 columnas, mientras que la máscara o kernel tiene un tamaño de 2 renglones por dos columnas. Cabe resaltar que este es solo un ejemplo y el tamaño del kernel puede variar pero se suele usar un tamaño de 3x3.

Con los tamaños de entrada 3×4 y de kernel de 2×2, tendremos una tamaño de salida, o mapa de características de 2×3. Es decir de 3 renglones y 2 columnas. Esto es así porque no se está considerando agregar un "relleno" pero en casos que se requiera mantener el tamaño de imagen de salida o "mapa de características", igual al tamaño de imagen de entrada se puede agregar un relleno. Para generar cada uno de los píxeles de salida, o

del mapa de características se va a recorrer el kernel sobre la imagen de entrada. Cada que se recorre o cada que está en una nueva posición, se genera un nuevo píxel de salida. Analizando ahora un caso puntual del primer píxel del mapa de características, vemos que es la sumatoria de los valores que resultan de la multiplicación de los píxeles de la imagen de entrada con los píxeles que se alinea del kernel. Para generar el siguiente píxel del mapa de características, se recorre el kernel, lo que lo alinea con otros valores de la imagen de entrada, de nueva cuenta se suman los valores de la multiplicación entre kernel e imagen de salida. Así sucesivamente se repite el proceso de recorrer el kernel sobre la imagen.

Las características del kernel, como su tamaño y valores van a determinar la "función" de la convolución. Pueden realizar cosas como detectar bordes verticales, difuminar o resaltar bordes, entre otras.

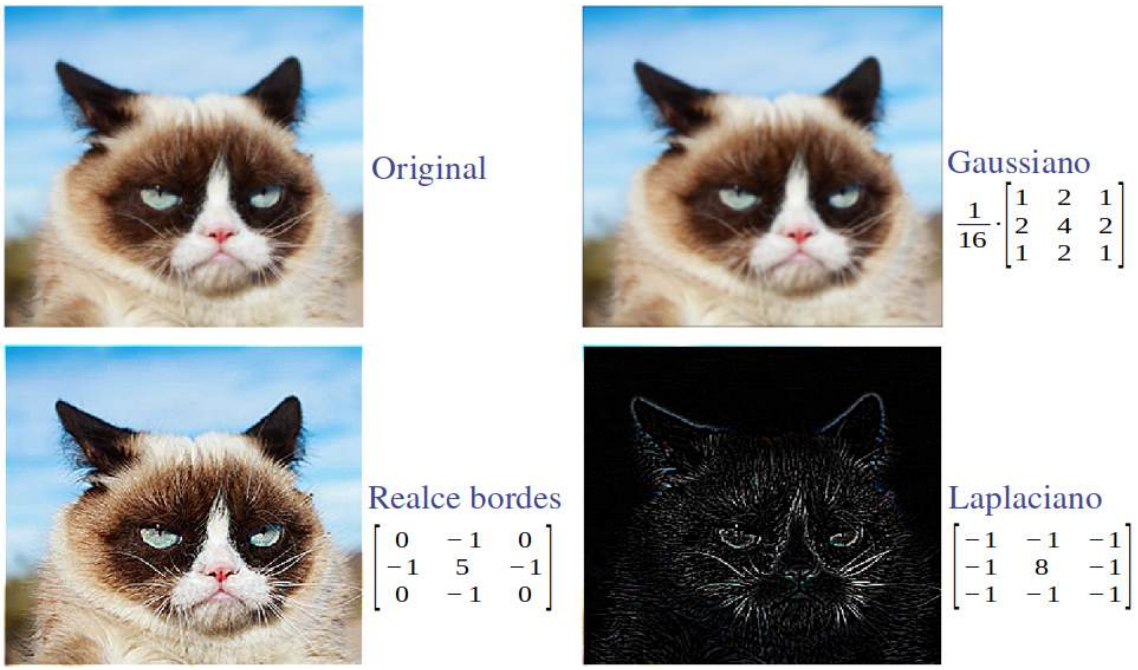


Figura 5.2. Diferentes tipos de kernel. Recuperado de [14].

En la figura 5.2 se muestran diferentes tipos de kernel, que realizan diferentes "funciones". En la parte superior izquierda se muestra la imagen original. Del lado superior derecho vemos un kernel de tipo Gaussiano. De forma visual se puede observar que el filtro Gaussiano realiza la función de "desenfoque" o de difuminación de los

bordes. También se puede observar que todos los valores de la matriz se dividen entre 16. Se observa también que el valor del centro del kernel es mayor al de los valores de alrededor. También los valores de las esquinas son menores. Este kernel es una especie de promedio entre el pixel que se va a analizar que sería el centro del kernel y los píxeles que están a su alrededor. Es el promedio por que si se suman todos los valores del kernel la suma de 16 y luego toda la matriz se divide entre 16. Pero como vimos la importancia aumenta en el centro del kernel.

Ahora pasamos a ver el caso del kernel del "realce de bordes" donde vemos que los pesos son negativos en la "cruz" y altos en el centro con valor de 5. Supongo el caso en que todos los valores que empatan con el kernel son más o menos similares. En ese caso al multiplicar los valores de la cruz por -1 y luego el valor del centro por 5, y hace la sumatoria tendremos que se resta 5 - 4 por lo que el valor del píxel resultante solo se multiplica por 1 es decir para igual. Ahora suponiendo que el valor del centro cae en un píxel que es un borde. Supongamos que ese borde es vertical y más oscuro que sus alrededores. En ese caso los valores cercanos al borde se van a aclarar y cuando el valor caiga directamente en el borde se van a oscurecer. Ya que los valores cuando esté cerca del borde van a generar valores muy altos, pero justo cuando caiga en el borde van a tener un valor muy bajo o "oscuro" [14].

5.2. Redes Neuronales convolucionales

En el caso de las redes neuronales convolucionales podemos tener una gran cantidad de kernels lo cual dará lugar a una cantidad similar de mapas de características, también se pueden ir realizando capas de convoluciones adicionales lo cual incrementará la profundidad de la red. También al final de las capas convolucionales se suelen agregar capas densas o "completamente conectadas", las cuales van a analizar o procesar los mapas de características y por ejemplo realizar la clasificación de la imagen.

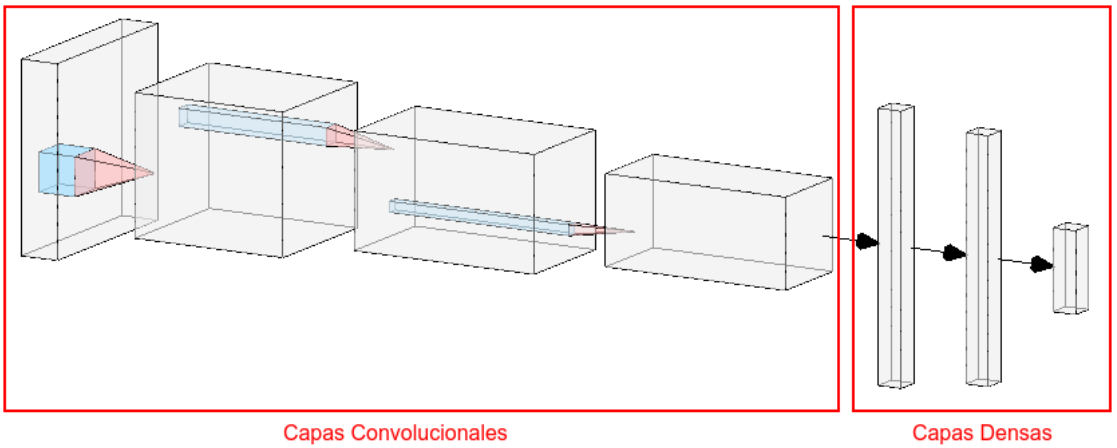


Figura 5.3. Ejemplo Sencillo de una red neuronal. Recuperado de [14].

En la figura 5.3 vemos un ejemplo sencillo de una red neuronal convolucional. Como se comentó las redes no solo es aplicar un filtro a una imagen si no una serie de filtros aplicados en diferentes etapas las cuales suele extenderse en cantidad de mapas de características. Es decir, cada vez que se aumenta en profundidad una red se suele aumentar la cantidad de filtros, y por lo tanto la cantidad de kernels.

5.3. Relleno de Borde.

Figura 5.4. Ejemplos de relleno de borde. Recuperado de [14].

Como se mencionó anteriormente cuando se realiza una convolución en una imagen, el tamaño de la imagen se reduce. Según la aplicación que se tenga este “efecto” puede

afectar el resultado. Es por esta razón que existen formas de generar el tamaño de salida igual al de la entrada.

En la figura 5.4 vemos ejemplos de relleno de borde, una de las más usadas es rellenar con ceros, o algún valor constante, esto no afectará mucho a la imagen resultante ya que es solo un relleno. Otra técnica es replicar los valores del borde es decir recorrer los valores del borde en el exterior. En el caso de las esquinas exteriores se vuelve a repetir el valor de la esquina interior. Otra técnica es la de "espejo" donde se toman los valores de "cruz" central y se usan para rellenar los bordes.

5.4. Desplazamiento o "stride".

Como ya se mencionó para generar el mapa de características se debe ir recorriendo el kernel sobre la imagen de entrada, pero este desplazamiento aunque suele ser de 1 no siempre es así.

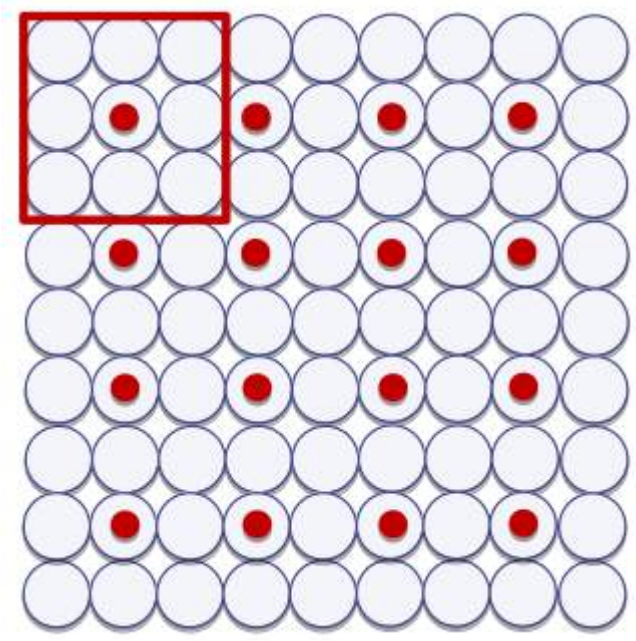


Figura 5.5. Desplazamiento del kernel de dos casillas. Recuperado de [14].

Una de las principales razones para usar un paso mayor a 1 es reducir la cantidad de datos. Por ejemplo en la figura 5.5 vemos una imagen de entrada de 9×9, lo cual daría

una salida de 7×7, usando un kernel de 3×3, y un paso de 1. Pero en este caso se va a usar un desplazamiento de 2 que irá centrado en los puntos rojos y que va a generar una salida de 4×4 en comparación de la salida de 7×7 lo cual reduce en gran medida la cantidad de datos y lo puedo hacer más fácil de procesar.

5.5. Submuestreo

Después de aplicar la convolución un paso o capa muy común en una red neuronal convolucional es el submuestreo. Básicamente se trata de tomar los valores de una determinada zona y aplicar alguna operación, para generar nuevos valores. Existen varios tipos de submuestreo como: máximo, promedio, L^2, entre otros. Por ejemplo el caso de submuestreo máximo de una zona de 2×2 toma el valor máximo de esta zona de 2×2. La idea general del submuestreo es poder hacer que la red sea resistente a variaciones de desplazamiento de la entrada. Hay características que van a ser relevantes independientemente de la posición de la imagen donde esté, lo cual puede determinar la clasificación.

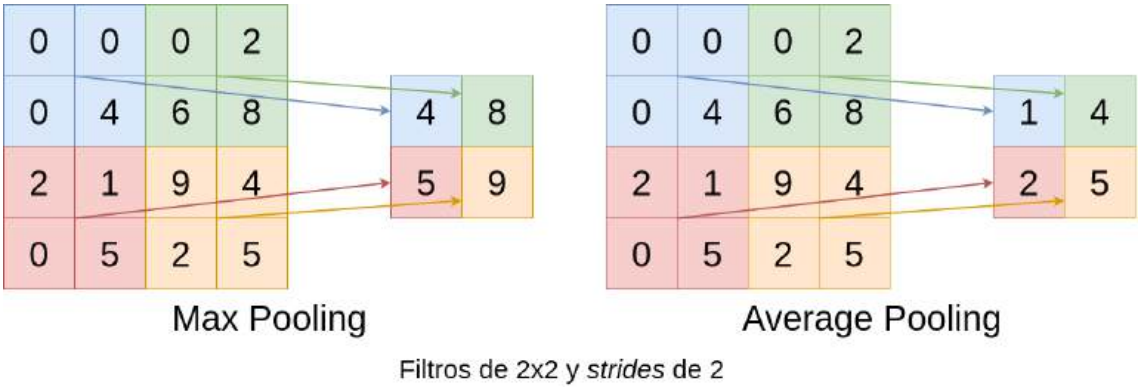


Figura 5.6. Tipos de submuestreo. Recuperado de [14].

Algunos de los tipos de submuestreo más comunes son el submuestreo máximo y el submuestreo promedio como se muestra en la figura 5.6. En el caso de muestreo máximo vemos un ejemplo que el filtro tiene un tamaño de 2×2 es decir que cada dos casillas se va a analizar los valores y se va a tomar el valor máximo por ejemplo en el

calor verde tenemos calores de 0, 2, 6 y 8 por lo que el valor de salida será de 8. Esto también reduce la salida, por ejemplo vemos que la imagen de entrada es de 4×4 con un paso de 2, lo cual generará una salida de 2×2. En el caso del submuestreo promedio se suman los valores entre sí y se dividen entre el número de casillas en este caso 4. Siguiendo con el caso del sector amarrillo la suma será de 16, dividido entre 4 el resultado es de 4. Como vemos los valores resultantes son muy diferentes y depende de la aplicación que tipo de submuestreo usar, pero uno de los más usados es el submuestreo máximo.

5.6. Algebra lineal

Una de las entradas del proceso, como se ha mencionado, son los archivos de mallas de polígonos en formato STL. Estos archivos básicamente son una lista de triángulos. Los triángulos están definidos por las coordenadas de 3 puntos en el espacio, y un vector normal. Importante mencionar que el vector normal define la parte exterior del objeto 3D. En otras palabras apunta hacia afuera del cuerpo. Para realizar varios de los análisis implementados en este proyecto se utilizaron.

5.6.1. Vector normal

Un vector normal es un vector que es perpendicular a un plano. Es decir que forma un ángulo de 90 grados. Esta definición también se puede ampliar a una superficie no plana. Donde el vector normal será un vector que es perpendicular a la superficie en un solo punto en específico.

Para los fines de este análisis, ya que los triángulos son superficies planas, el vector normal puede partir de cualquier punto del triángulo.

Es importante también mencionar que para el contexto del análisis el vector normal va más enfocado en la orientación de la cara.

5.6.2. Producto punto

El producto punto también conocido como producto escalar, es una operación en álgebra lineal que resulta de multiplicar cada uno de los componentes de un vector por

los componente de otro vector y realizar la sumatoria. Esto se puede expresar con la siguiente ecuación 5 donde *a* y *b* son vectores de *n* números de elementos [15].

$$a \cdot b = \sum_{i=1}^{n} a_i b_i \tag{5}$$

5.6.3. Producto Cruz

El producto cruz es un tipo de producto vectorial que solo está definido en R^3. Supongo que u y v son vectores de R^3 de la siguiente forma. $u = a_1 i + b_1 j + c_1 k$ y $v = a_2 i + b_2 j + c_2 k$. Entonces el producto cruz está definido por la ecuación 6 que se muestra a continuación [15].

$$u \times v = (b_1 c_2 - c_1 b_2) i + (c_1 a_2 - a_1 c_2) j + (a_1 b_2 - b_1 a_2) k \tag{6}$$

5.6.4. Producto Mixto

Resulta útil para revisar si 3 vectores son coplanares utilizar el producto mixto el cual nos dará 0 en el caso que los tres vectores son coplanares.

Se define el producto mixto como el producto cruz entre los primeros dos vectores seguido por el producto punto entre el vector resultante y el tercer vector.

Tenemos la ecuación 7 que lo describe donde u, v, w, son vectores del espacio euclidiano 3 dimensional [16].

$$[u, v, w] = (u \times v) \cdot w \tag{7}$$

5.7. Métricas de evaluación

Se usan 3 métricas de evaluación primeramente el error absoluto medio (MAE por sus siglas en inglés) ya que guarda relación con las unidades que se están midiendo, descrita en la ecuación 8, donde x_i es un valor objetivo o referencia, y_i son las predicciones, y *n* es el número de datos.

Después para revisar qué tanto está desviado el error en cada una de las predicciones de forma individual, se usa la desviación estándar DS aplicado al error MAE, descrita en la

ecuación 9, donde x_i es un valor del conjunto, en este caso el MAE de un dato individual, $\bar{x}$ es la media, en este caso el MAE global, y n es el la cantidad de datos.

Ya que las coordenadas se usan para reconstruir el polígono o contorno de la pieza se usa la intersección sobre unión (IoU por sus siglas en inglés) que mide una relación entre los datos objetivo y la predicción, se calcula al dividir el área de intersección, entre el área unión. Si las áreas coinciden perfectamente se tiene un valor de IoU de 1, se describe en la ecuación 10, donde A es el área de la etiqueta o área objetivo, y B es el área de predicción.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \tag{8}$$

$$DS = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}} \tag{9}$$

$$IoU = \frac{A \cap B}{A \cup B} \tag{10}$$

5.8. Características y ventaja del Sketch

Los dibujos paramétricos o sketch tienen varias características las cuales presentan varias ventajas en la tarea de definir una geometría por ejemplo poder agregar restricciones geométricas y restricciones dimensionales.

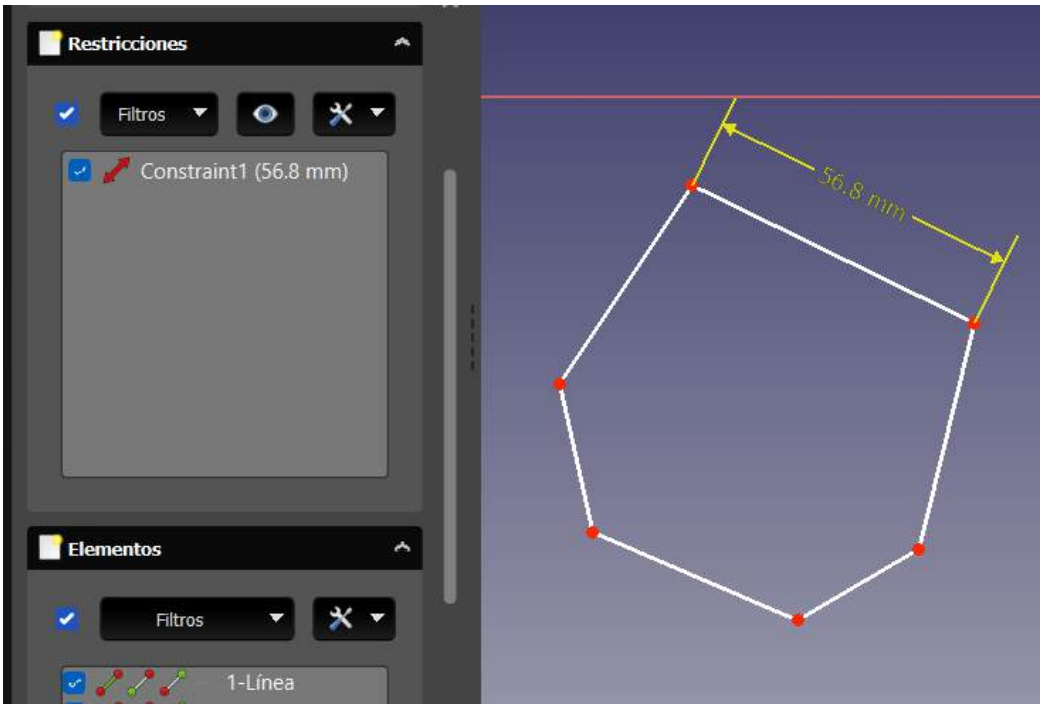


Figura 5.7. Aplicación de restricciones dimensionales, en este caso de longitud.

En la figura 5.7 vemos la aplicación de una restricción dimensional a una de las líneas del contorno. En este caso se aplica una restricción de longitud de 56.8mm la cual puede variar según las necesidades del diseño.

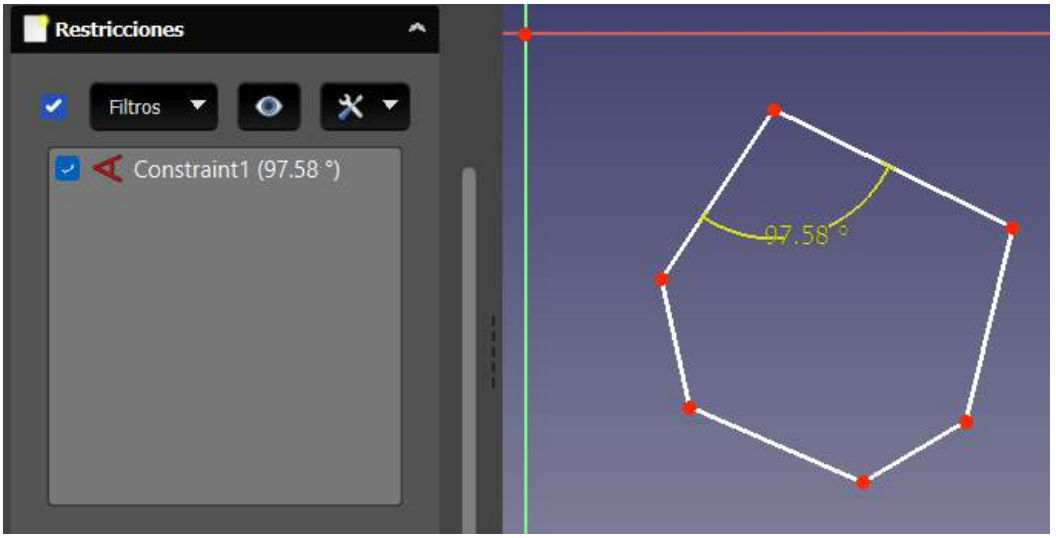


Figura 5.8. Restricción dimensional de ángulo.

Siguiendo con las restricciones dimensionales en la figura 5.8 vemos la aplicación de la restricción de la dimensión del "ángulo" entre dos de las líneas del contorno. De igual manera esta restricción de 97.58° grados se puede observar en la lista de restricciones en la parte izquierda.

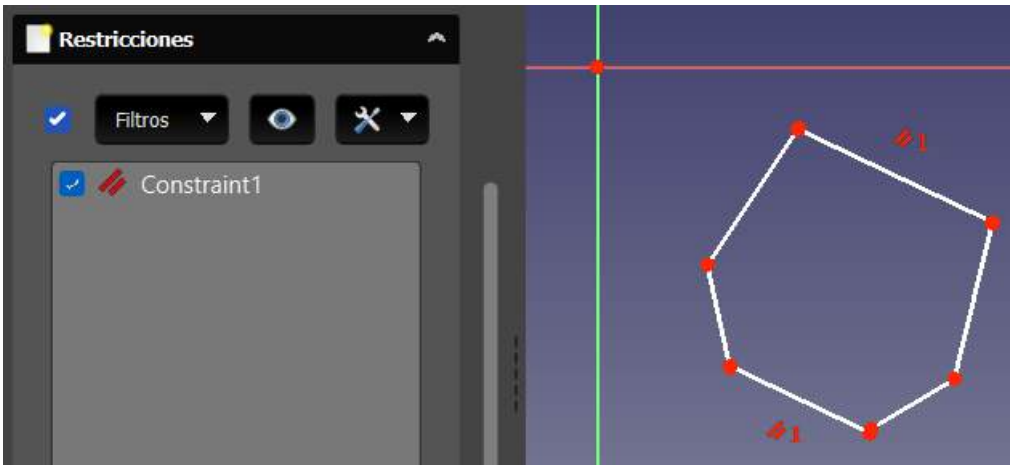


Figura 5.9 Restricción Geométrica. Paralelismo.

Pasando al apartado de restricciones geométricas, es decir que describen las relaciones entre geometrías. Tenemos el caso de la figura 5.9, en la cual se aplica la restricción geometría de paralelismo entre dos de las líneas de control. Esta restricción puede ser observada en la lista de restricciones en la parte izquierda de la figura.

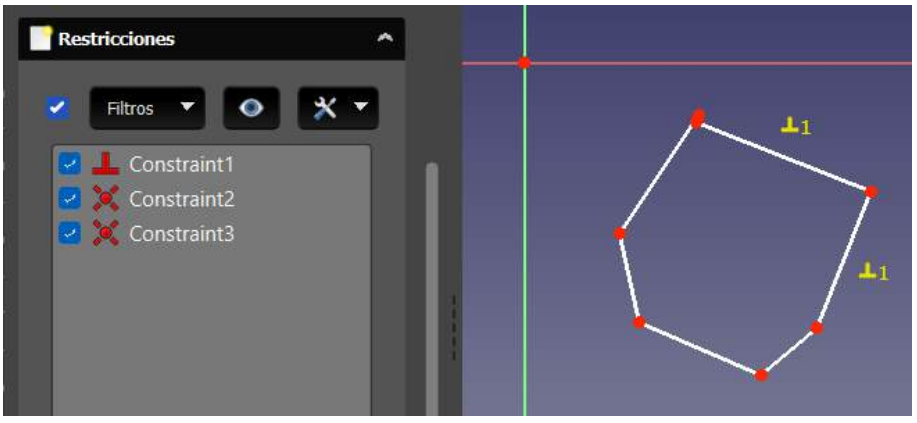


Figura 5.10. Restricciones geométricas. Perpendicularidad y coincidencia.

Otra restricción geométrica que se puede aplicar es la de perpendicularidad, lo cual se observa en la figura 5.10, en ella se aplica perpendicularidad entre las dos líneas del contorno situadas en su parte superior derecha. Se puede apreciar que la geometría cambia ligeramente. En esta ocasión se agrega también dos restricciones de coincidencia para que la líneas inicien donde termina la otra. Esto nos ayuda a asegurar que el perímetro es constante o continuo. En otras palabras que no se separa o se parte el contorno de la figura. De igual manera estas tres restricciones se observan enlistadas en la parte izquierda, lo que permite hacer seguimiento o referencias entre ellas.

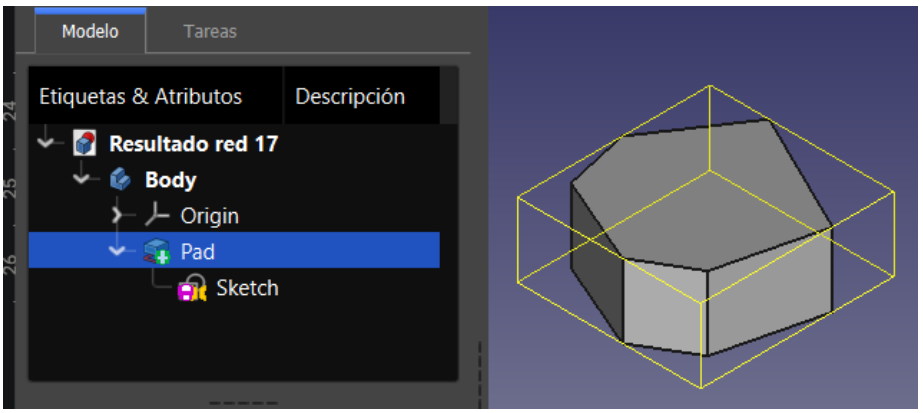


Figura 5.11. Operación de extrusión.

Ahora en la figura 5.11 vemos como al sketch se le aplica la operación volumétrica. En este caso las operaciones de extrusión. Aquí vemos cómo se genera un volumen que levanta el área del sketch en hacía arriba en el sentido positivo del eje z.

Es posible también agregar otro tipo de características a la operación de extrusión por ejemplo un ángulo de extrusión que es útil para aplicaciones como piezas que van a ir en un molde de inyección.

También en otras aplicaciones se pueden agregar extrusiones en dos direcciones lo cual puede resultar útil para aplicaciones de fundición. Donde el sketch sirva como línea de partición en un solo plano.

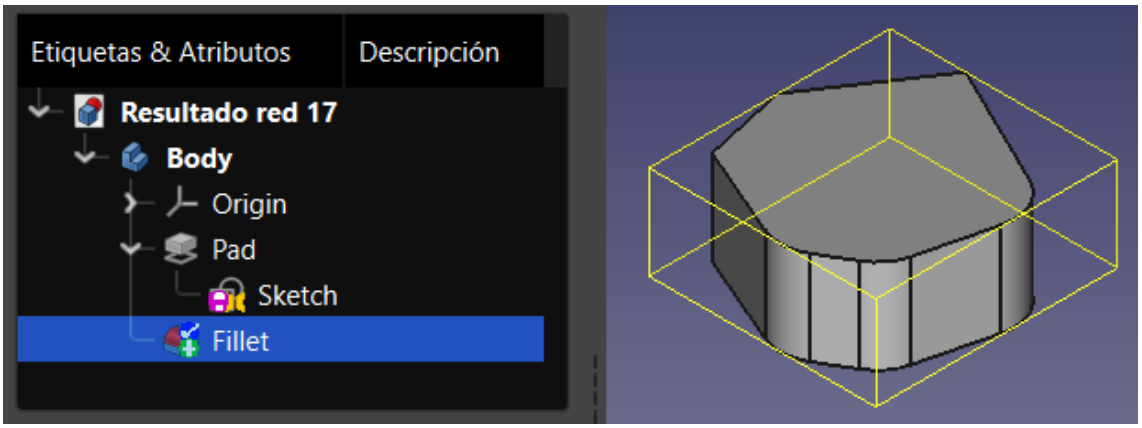


Figura 5.12. Aplicación de operación volumétrica de Redondeo.

En la figura 5.12 vemos como se puede aplicar al sólido generado un refinamiento de la forma, que en esta caso es las esquinas modificarlas para agregar redondeos, los cuales permiten tener geometrías más refinadas. Según sea el caso los redondeos son muy útiles ya que en algunas ocasiones es complicado generar esquinas rectas por ejemplo en el caso de maquinado por lo que zonas redondas son más sencillas de maquinar. También si la pieza va a tener aplicaciones donde reciba mucha fuerza el uso de redondeos ayuda a distribuir los esfuerzos y evitar tener puntos de falla.

6. Hipótesis

Es posible recuperar las coordenadas de los vértices del contorno desde un archivo de mallas de polígonos o desde una imagen usando redes neuronales convolucionales. Posteriormente usar estas coordenadas de vértices para generar los elementos geométricos (líneas rectas), en un script que se ejecute para reconstruir el sketch en la interfaz gráfica de usuario de FreeCAD.

7. Objetivos

7.1. Objetivo General

Reconstruir el sketch en la GUI de FreeCAD, a partir de archivos de mallas de polígonos. Se procesan los archivos de mallas de polígonos para generar imágenes de contornos, posteriormente usar redes neuronales convolucionales para extraer las coordenadas. Con las coordenadas extraídas reconstruir el sketch usando el software de código abierto FreeCAD.

7.2. Objetivos Específicos

- Procesar mallas de polígonos para generar una imagen de la cara principal con el contorno exterior identificado.
- Generar un base de datos sintética de contornos compuestos de líneas rectas, los contornos están compuestos por máximo 6 líneas rectas.
- Extraer las coordenadas de los vértices del contorno con una red neuronal convolucional.
- Usar las coordenadas para generar las líneas rectas en un script y ejecutarlo en FreeCAD para generar el sketch en la GUI de FreeCAD.

8. Limitaciones

Como limitación del alcance del proyecto se propone usar imágenes generadas de forma sintética. Posteriormente el alcance se podría extender a usar imágenes reales, pero por el momento queda fuera del alcance. Una de las principales razones para usar datos sintéticos es, la gran cantidad de datos que requieren las redes convolucionales para entrenar. Otra limitación es usar imágenes exclusivamente compuestas de líneas para simplificar el análisis dejando fuera curvas o algún otro tipo de geometría primitiva o paramétrica. Para simplificar el análisis por limitaciones de capacidad de cómputo se propone usar imágenes pequeñas de 128 por 128 píxeles. Para ajustarse a la salida de

tamaño fijo de la red convolucional, se propone usar imágenes de entre 3 y 6 líneas, y de esta forma tener siempre una salida de 6 elementos por 2 componentes x, y, para la salida de la red.

9. Metodología

El objetivo de este trabajo es reconstruir el sketch o "dibujo paramétrico", partiendo de archivos "no paramétricos". Durante el proceso de investigación realizado en este proyecto se logran identificar 2 métodos para reconstruir el sketch. La principal diferencia entre estos dos procesos es el punto de partida, uno parte de mallas de polígonos, y el otro de imágenes del contorno de objetos 3D.

Para reconstruir el sketch o dibujo paramétrico, es necesario extraer o reconstruir los parámetros que describe al dibujo, esto puede simplificar se como extraer las coordenadas de los vértices del contorno de una figura. Recordemos que como limitaciones de este trabajo sólo se analizan contornos compuestos únicamente por líneas rectas.

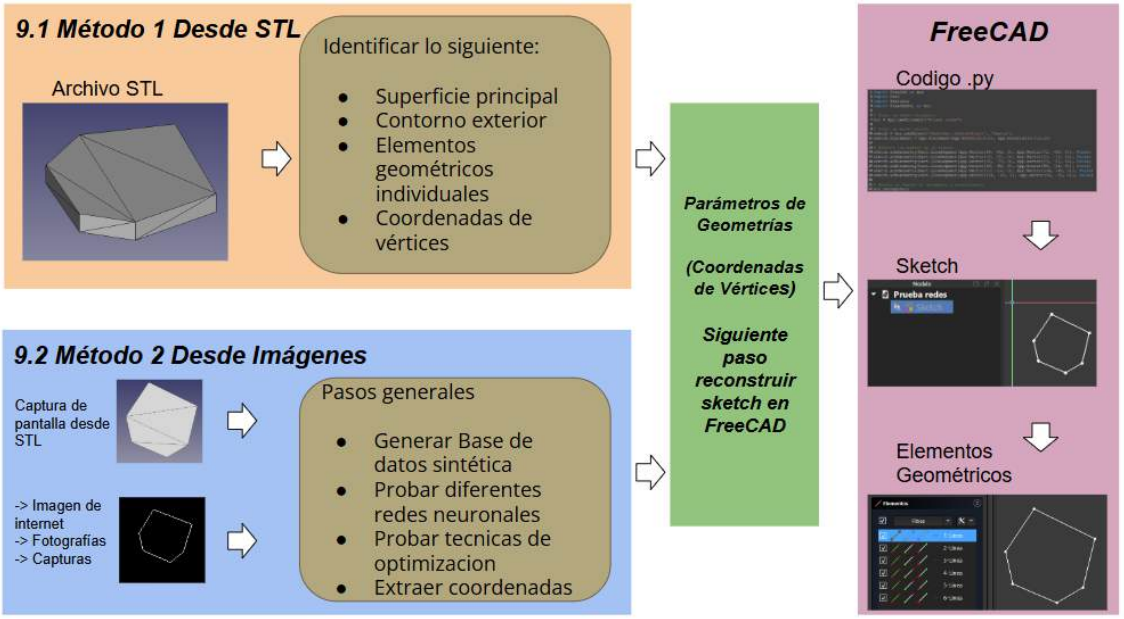


Figura 9.1. Métodos para la reconstrucción del Sketch.

Para ilustrar estos procesos vemos en la figura 9.1, los procesos propuestos de manera general, la principal diferencia son el tipo de archivo de entrada. El método 1 toma como entrada archivos de mallas de polígonos en formato STL. Por otro lado el método 2 toma como entrada imágenes, que pueden provenir de diferentes fuentes por ejemplo fotos, imágenes sintéticas, capturas de pantalla que se pueden generar desde archivos STL. Se puede considerar al método 2 como una extensión del alcance ya que puede analizar archivos de mallas de polígonos al generar una captura de pantalla del STL, así como imágenes de otras fuentes. También en el método 2 se puede ingresar un STL de forma directa, y un preprocesamiento automático se encarga de encontrar la cara principal, identificar el contorno exterior, para generar una imagen del contorno exterior.

Ambos procesos buscan encontrar las coordenadas de los "vértices" de las figuras analizadas, lo cual se muestra en el bloque verde. Posteriormente en el bloque rosa con estas coordenadas se generan líneas de código para cada elemento geométrico, se puede entender que es el sketch en forma de código, ya que es archivo formato .py o un script que describe al sketch o dibujo paramétrico. Este script después se ejecuta en el software de diseño mecánico FreeCAD, lo cual a su vez va a generar un archivo de FreeCAD que va a contener el sketch o dibujo paramétrico que se desea reconstruir. Por último este archivo de FreeCAD que contiene al sketch, dentro del sketch tiene cada uno de los elementos geométricos identificados.

9.1. Reconstrucción del sketch a partir de mallas de polígonos

En este apartado se van a usar como punto de partida, archivos de mallas de polígonos en específico en formato STL. Se van a analizar contornos que están compuestos por líneas rectas y por fragmentos de arcos. En el siguiente apartado, que realiza la reconstrucción del sketch desde imágenes, se limita a solo usar líneas rectas.

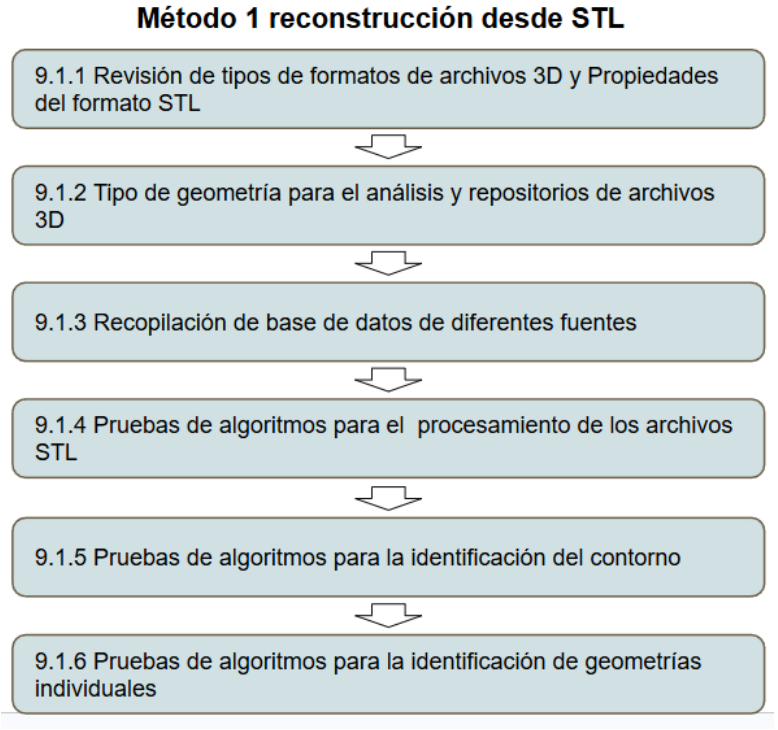


Figura 9.2. Método 1 reconstrucción del sketch desde STL.

A continuación se describe a grandes rasgos el método 1 para reconstruir el sketch desde STL se muestra un diagrama en la figura 9.2. Primero se analizaron los tipos de "formatos" archivos 3D a utilizar en este caso se analizaron archivos en formato IGES, STEP, y STL, ya que son algunos de los formatos más utilizados para exportar archivos 3D de diseño mecánico. Se seleccionan archivos STL por su amplia disponibilidad y su formato simple. Se hace un análisis de los componentes del archivo STL para plantear formas de procesarlo. Se definió el alcance en el tipo de geometrías que se pueden analizar, se acota a geometrías regulares, definidas por superficies planas, y que el contorno que se va a analizar esté compuesto exclusivamente por líneas rectas y arcos (como aclaración en el método 2 sólo se analizan líneas rectas). Con el tipo de formato y el tipo de geometrías definidos, se buscan repositorios que contengan dichas características, se utilizan principalmente 2 repositorios thingiverse y grabcad, por la gran cantidad de archivos STL que contienen, así como por sus licencias de uso para fines "no comerciales". Los datos se recopilan de forma manual de los repositorios, y algunos archivos son creados de autoría propia. Se hacen pruebas para encontrar

algoritmos con el fin de procesar los archivos STL, lo cual abarca encontrar la superficie principal, identificar el contorno exterior de la superficie. Luego se realizan pruebas para identificar las geometrías, los cual abarca analizar el contorno para subdividir los elementos geométricos individuales, y luego extraer sus parámetros o en otras palabras las coordenadas de los vértices.

9.1.1. Revisión de tipos de formatos de archivos 3D y Propiedades del formato STL

El problema de intercambiar objetos 3D entre diferentes tipos de software ha sido solucionado de varias maneras. Existen varios tipos de formatos. En 1979 en la reunión del Departamento de defensa en Detroit, se planteó la necesidad del intercambio de información entre sistemas de diseño asistido por computadora (CAD por sus siglas en inglés) y sistemas de manufactura asistida por computadora (CAM por sus siglas en inglés), lo cual llevaría a la creación de la especificación inicial de intercambio de gráficos (IGES por sus siglas en inglés) [17].

Posteriormente en 1994 surge ISO 10303-21 que especifica un formato de intercambio, normalmente conocido como Estándar para el intercambio de datos del modelo del producto (STEP por sus siglas en inglés), que permite que los datos del producto conforme a un esquema en el lenguaje de modelado de datos EXPRESS (ISO 10303-11) se transfieran entre sistemas computacionales [18]. Algunas de las ventajas de STEP sobre IGES fueron descritas en [19] y son:

1.- Permite la visualización y modificación de geometría utilizando cualquier herramienta CAD capaz de interpretar la geometría STEP y rompe la dependencia entre los sistemas CAD y la definición del producto.

2.- Muy útil para agrupar elementos mecánicos en determinada vista.

3.- Puede incluir en el esquema de montaje, todas las definiciones de conexión.

4.- Facilidad de derivación del contenido de información implícita.

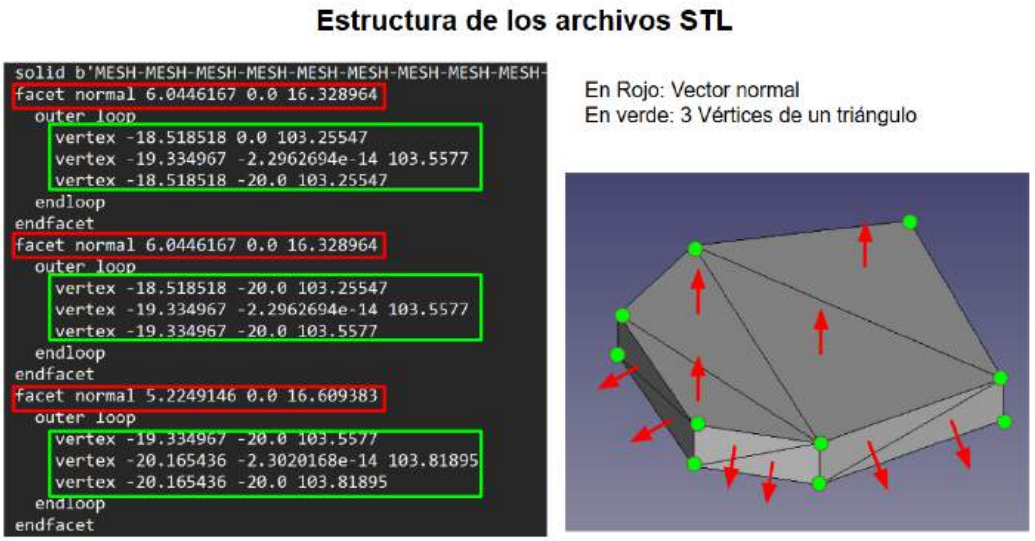


Figura 9.3. Estructura de los archivos STL.

El formato STL fue creado por 3D Systems Inc. Puede ser especificado como representación texto en formato de código americano estándar para el intercambio de información (ASCII por sus siglas en inglés) y binario, siendo los binarios más compactos. Un archivo STL describe una objeto 3D usando triángulos. No hay información de la conexión entre los triángulos. Los triángulos están definidos por un vector normal y los 3 vértices del triángulo (ordenados por la regla de la mano derecha) [20]. Se compone de las caras (facet) de triángulos como un conjunto de tres vértices y un vector normal. El archivo no contiene información topológica, de cómo están unidos entre sí los elementos, en el caso de los vértices, sus coordenadas se repiten tantas veces como aparezcan en la malla [21]. En la figura 9.3 se muestra la estructura de los archivos STL.

Como se mencionó se opta por utilizar el formato STL por su simpleza y su gran disponibilidad. Es un formato de objeto 3D que tiene muchas aplicaciones y se puede generar a partir de escaneos, nubes de puntos, diseños mecánicos así como estar disponible en muchos repositorios de impresión 3D.

9.1.2. Tipo de geometría para el análisis y repositorios de archivos 3D

Existen varios tipos de objetos 3D, se pueden clasificar según su forma en objetos regulares e irregulares. En el diseño mecánico se suelen usar formas regulares por ejemplo líneas rectas, superficies planas, redondeos de radio constante, arcos, círculos entre otras figuras geométricas relativamente simples. En contraste para otros tipos de aplicaciones como diseño gráfico se suelen utilizar geometrías irregulares por ejemplo de figuras orgánicas como animales y plantas, así como figuras para escultoras o alguna otra forma que suele ser irregular. En la figura 9.4 podemos ver un ejemplo de formas regulares e irregulares.

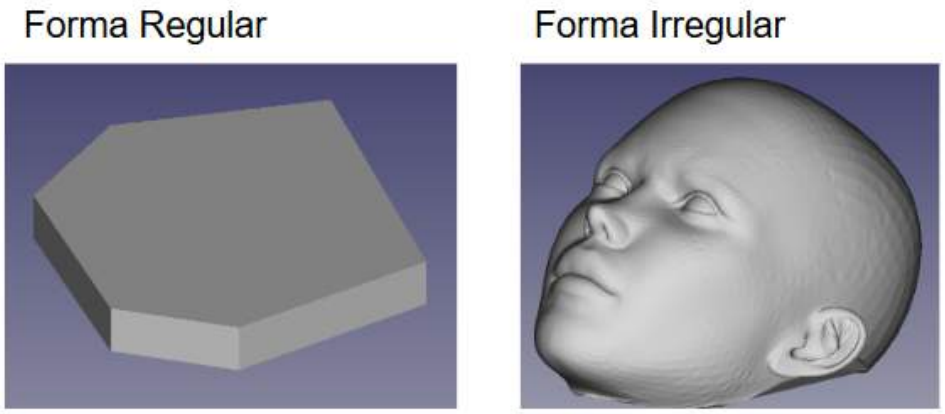


Figura 9.4. Formas regulares e irregulares.

Además de la cara plana otra consideración importante es un contorno simple. Se buscan archivos en los cuales el "contorno" de la superficie principal esté compuesto en exclusivamente por líneas rectas y en el caso del método 1 también de arcos. Esta restricción es para poder simplificar el análisis quedando fuera otro tipo de geometrías como curvas tipo splines, como se muestra en la figura 9.5, donde de lado izquierdo vemos una forma que tiene una cara plana pero su contorno contiene muchas curvas complejas las cuales son complicadas de parametrizar. Por otro lado en la parte derecha vemos una forma cuyo contorno está compuesto exclusivamente por líneas rectas, como se remarca en rojo.

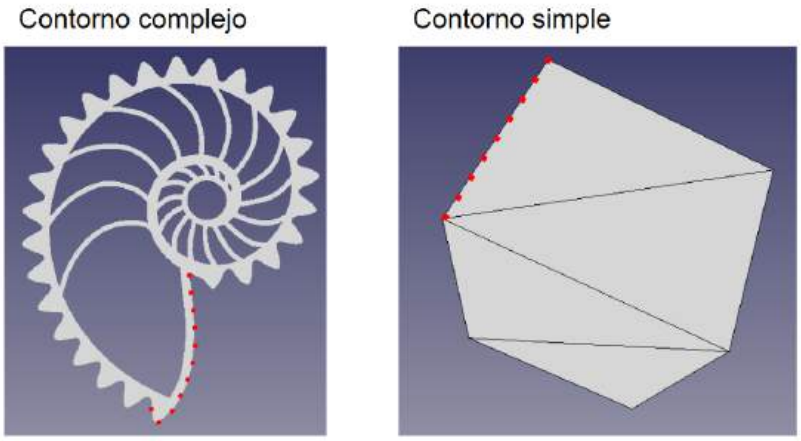


Figura 9.5. Contornos simples y complejos.

Para encontrar estos archivos se recurrió a varias fuentes una de las principales fueron los repositorios 3D, existen varios repositorios en internet que contiene archivos 3D. Para recopilar los archivos se usaron principalmente 2 repositorios thingiverse y grabcad cuyos logos se muestran en la figura 9.6. Ambos repositorios cuentan con uso libre para los modelos disponibles para fines no comerciales, por un lado los modelos de thingiverse se distribuyen bajo de licencia de "comunes creativos" y por otro lado lo archivos de grabcad en sus términos y condiciones en el apartado sexto de "archivos subidos por usuario", se estipula que los archivos usados pueden usarse libremente para usos no comerciales.

Figura 9.6. Repositorios de archivos 3D.

9.1.3. Recopilación de la base de datos

Una de las ventajas de thingiverse es la gran cantidad de archivos que existen, sin embargo las consideraciones de los tipos de archivos que vamos a recopilar hizo necesario que se usarán también los archivos de grabcad ya que en este repositorio la mayoría de los objetos 3D son de diseño mecánico lo cual simplificó la adquisición.

La adquisición de estos archivos fue la descarga de forma manual de cada uno de los archivos este método se utilizó para poder preseleccionar los archivos y descartarlos según las características que se han descrito como lo son una cara plana y un contorno simple.

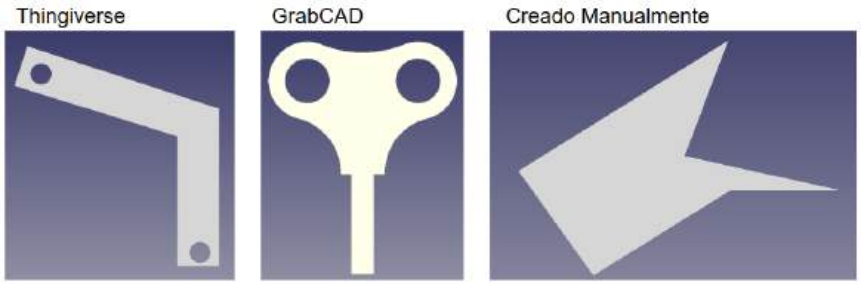


Figura 9.7. Archivos de la base de datos en formato STL.

Si bien en ambos repositorios existe una gran cantidad de archivos las limitaciones del presente proyecto son considerables lo que llevó al descarte de muchos archivos. Para suplementar este descarte se recurrió a generar archivos de forma "manual". Como ejemplo en la figura 9.7 se muestran archivos de la base de datos, de izquierda a derecha vemos un archivo tomado de thingiverse, luego uno de grabcad y por último uno de autoría propia generado "manualmente". Entre todas estas fuentes se logró conseguir aproximadamente unos 102 archivos para el análisis.

9.1.4. Procesamiento del STL

Para procesar los STL uno de los primeros pasos es identificar la cara o superficie principal. Como ya comentamos los tipos de geometrías que se van a analizar pasaron por un filtro, en el que tienen caras planas y contornos compuestos por líneas rectas y solo para el método 1 también de arcos.

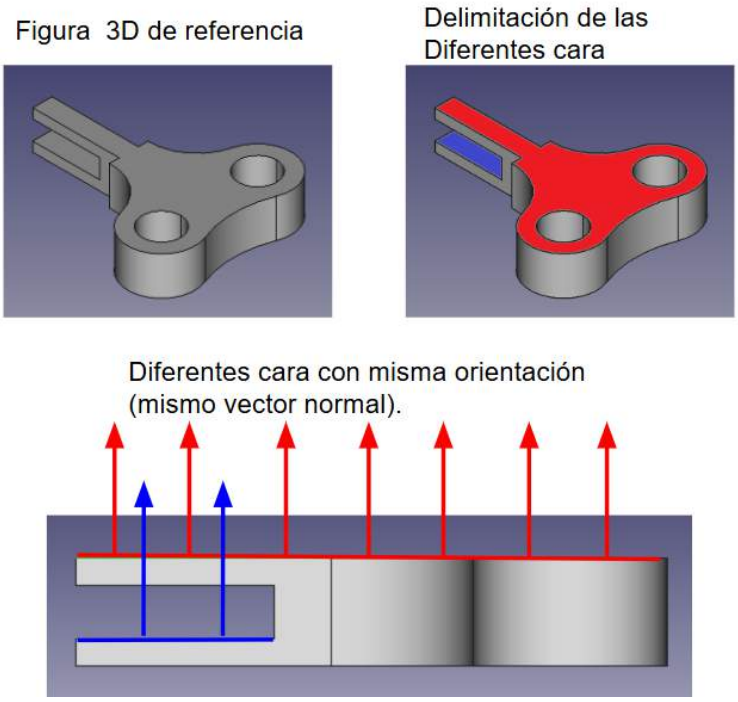


Figura 9.8. Diferentes caras en un STL, con la misma normal.

Recordando que la estructura de un STL es solo una lista de triángulos, y cada triángulo está definido por tres vértices y un vector normal. En el STL no hay información de cómo se conectan los triángulos entre sí.

Para identificar las caras individuales en un STL se puede analizar el vector normal que determina la orientación por lo que triángulos con el mismo vector normal tienen la misma orientación. Aunque esto por sí solo no es suficiente para determinar si estos triángulos son parte de la misma cara. Ya que puede ser el caso como se muestra en la figura 9.8 que dos caras diferentes tengan el mismo vector normal o orientación. Se observa en la parte superior izquierda el objeto 3D de referencia, y en la parte superior derecha dos caras marcadas en rojo y azul, las cuales tienen la misma orientación. Se puede ver en la parte inferior de la imagen como los triángulos que se encuentran en ambas caras tienen el mismo vector normal por lo que no está claro que son de diferentes caras, para ello se propone un análisis adicional.

Por lo que para identificar qué triángulos son parte de la misma cara propone realizar dos pruebas, por un lado el análisis del vector normal para su orientación y por otro lado

la confirmación de que estos triángulos son coplanares para asegurar que son parte de la misma cara. Con esto se puede diferenciar las distintas caras de un objeto 3D.

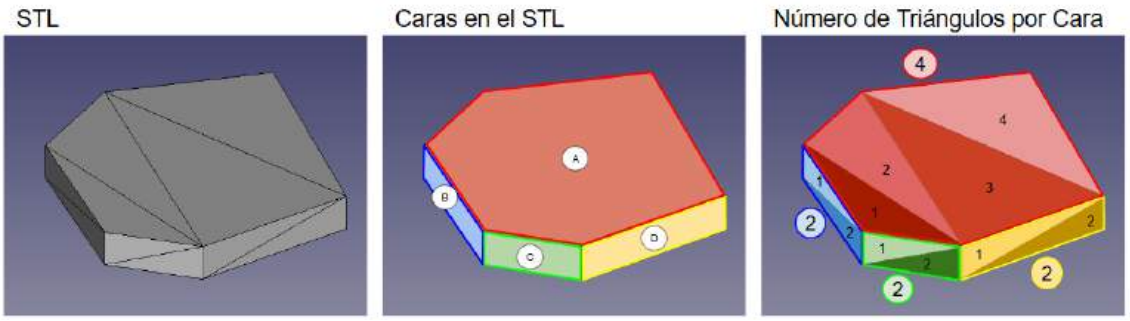


Figura 9.9. Identificación de cara principal en un STL.

Como se puede ver en la figura 9.9 un objeto 3D puede tener varías caras, incluso puede tener caras que se orientan en la misma dirección según su vector normal. Aquí la tarea es identificar la cara principal es decir la cara que aporta la mayor información del objeto 3D. Se puede entender que la cara principal es la que mayor cantidad de características geométricas tiene. Por ejemplo dado un STL que se muestra en el lado izquierdo, en la imagen podemos ver la cara A, B, C y D, remarcadas en colores distintos. Cada una de estas caras está compuesta por triángulos. Se propone la hipótesis de que la cara "principal" será la cara que tenga el mayo número de triángulos, por lo tanto tendrá el mayor número de características geométricas y es la cara que mayor información aporta para analizar el contorno del objeto 3D para poder extraer o reconstruir el sketch. Siguiendo el ejemplo de la figura tenemos que la cara A remarcada en rojo es la cara principal ya que como se muestra en la derecha esta cara está compuesta por 4 triángulos por lo que tiene mayor cantidad de "características geométricas".

9.1.5. Identificación del contorno

Una vez que la cara principal esté identificada, el siguiente paso será identificar el contorno exterior de esa cara. De nueva cuenta recordando que esta cara está compuesta solo por lo triángulo que a su vez están descritos solo por 3 vértices y un vector normal. También recordemos que un mismo vértice se puede "repetir" en varios triángulos. En otras palabras, un mismo vértice puede aparecer en varios triángulos.

Se propone como primer paso para encontrar el contorno, delimitar puntos máximos y mínimos que se realiza al tomar los valores máximos y mínimos de los componentes x, y de todas las coordenadas. En otras palabras el punto mínimo se determina al analizar todas las coordenadas y tomar el valor mínimo en x, luego agregarle el valor mínimo en y. Para los valores máximos se realiza lo mismo tomando el máximo en x, y el máximo en y.

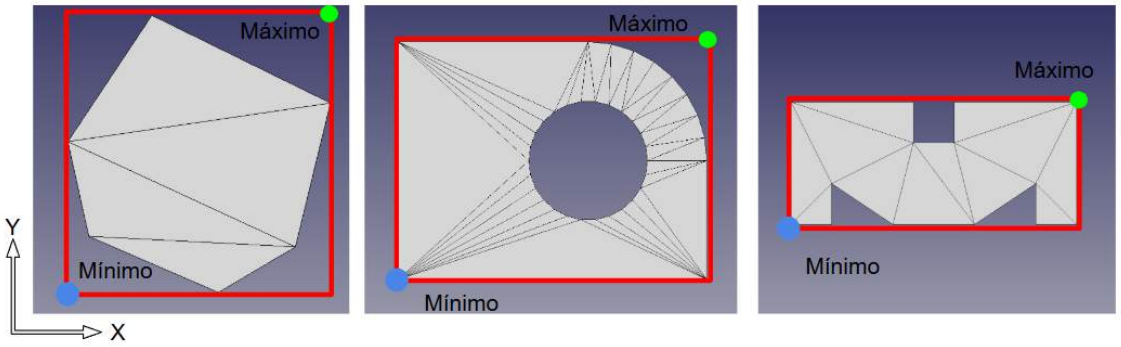


Figura 9.10. Puntos máximos y mínimos.

Un ejemplo de cómo se ven estos puntos máximos y mínimos se observa en la figura 9.10, donde se puede apreciar en azul el punto de referencia mínimo y en verde el punto de referencia máximo. Se observa también que no necesariamente estos puntos forman parte de los vértices de nuestra cara principal.

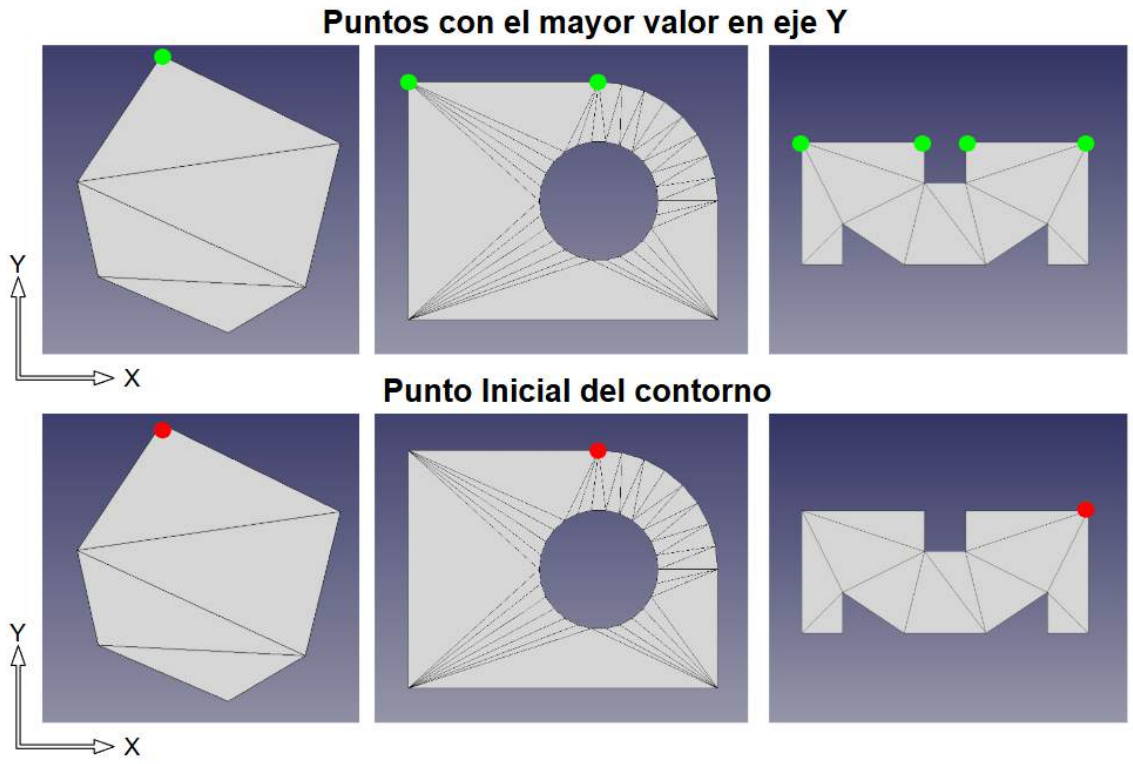


Figura 9.11. Determinación del punto inicial.

Ahora se necesita un punto inicial para tomar de punto de partida en el análisis de contorno. Lo que se propone es tomar un vértice de la parte superior derecha e ir bajando o "recorriendo el contorno". Se muestra una ilustración de este proceso en la figura 9.11. Primero se toman lo vertices con el valor máximo en el eje Y, estos vértices se muestran en color verde en la figura, en caso que existan varios vértices con el mismo valor en el eje Y se toma el que tenga el mayor valor en el eje X, los cual se muestra en rojo. De esta manera se selecciona el vértice que esté más cercano a la esquina superior derecha. Este proceso también nos asegura que no hay nada por arriba de este vértice.

Ahora se pasa al proceso de ir recorriendo o "bajando" por el contorno, para eso primero se identifican los vértices que se conectan al vértice inicial. Lo cual se muestra en la figura 9.12. En rojo se muestra el vértice inicial, y en amarrillo los vértices que están conectados a este. Como podemos ver ningún vértice está arriba del vértice rojo.

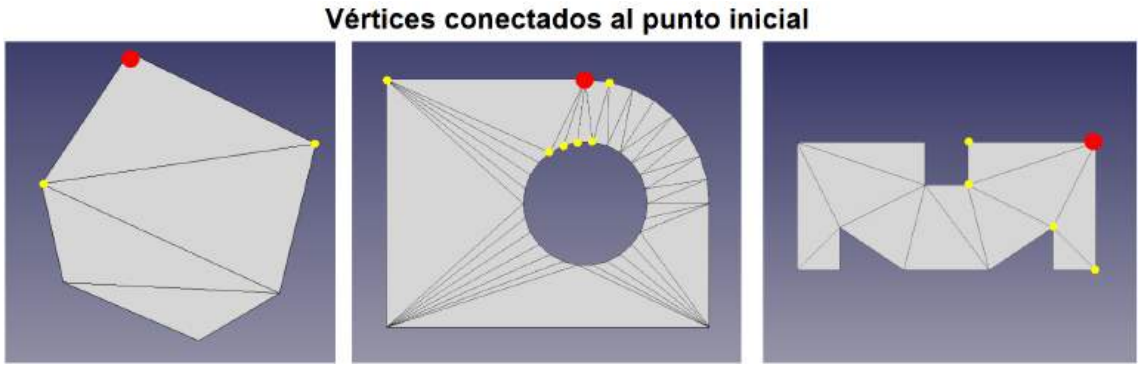


Figura 9.12. Vértices conectado al punto inicial.

En esta parte lo que se busca es encontrar el "contorno" exterior, pero recordemos que en el archivo STL no hay listas de líneas o aristas, únicamente de vértices y vectores normales. Por lo que las líneas están definidas con los vértices. En la figura 9.13 se muestran estas líneas que vamos a analizar para ver que "linea" es la siguiente en el control en sentido horario, o en otras palabra en dirección hacia abajo y la derecha.

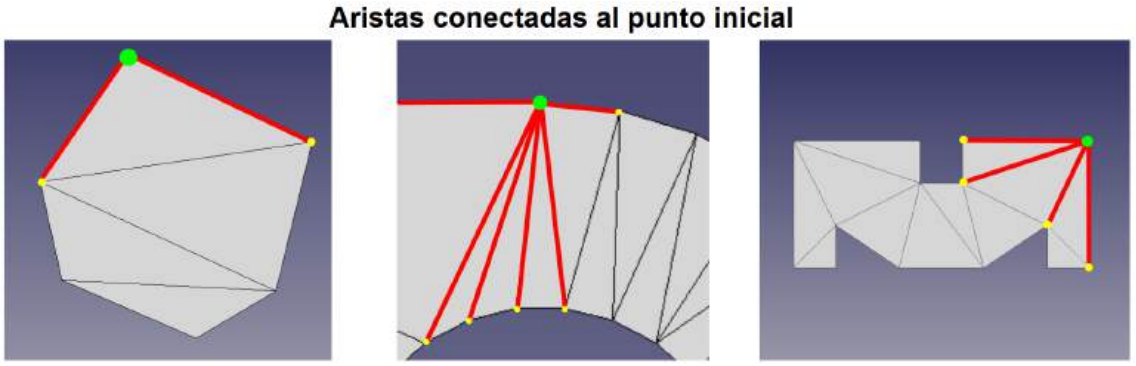


Figura 9.13. Aristas conectadas al punto inicial (vértices).

En la figura 9.14 vemos el proceso para determinar cuál es la siguiente línea en el contorno. Se analiza el ángulo que forma las líneas conectadas al punto inicial respecto a una línea auxiliar horizontal. La línea auxiliar horizontal se muestra en verde. El ángulo se mide en sentido horario, para todas las líneas, esto es importante ya que vamos a comparar los ángulos entre sí. Se muestra en azul los ángulos formados y en naranja el ángulo menor. La línea que forme el ángulo menor será la siguiente línea en cerrar el contorno, por que es la primera arista o línea que se puede "encontrar". La idea

detrás de este proceso es que el contorno se va recorriendo o censando desde afuera hacia adentro.

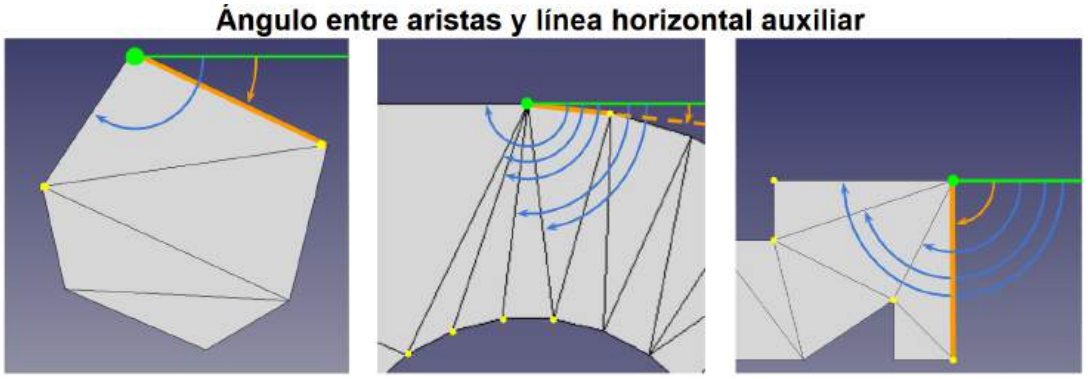


Figura 9.14. Revisión de ángulos respecto al vértice inicial.

Para encontrar la segunda arista, el proceso anterior se repite pero en esta ocasión se toma la última arista que se encontró como la arista inicial y a partir de aquí se miden los ángulos.

El concepto o la idea de ir recorriendo el contorno desde su parte exterior se ilustra en la figura 9.15. La idea es que se pueda ir cerrando un ángulo desde la última arista o la arista previa para poder encontrar desde afuera cual es la siguiente arista, esta es la idea o el concepto que se propone para recorrer el contorno. Para lograr este objetivo se busca ir comparando los ángulos entre las aristas para encontrar el menor y así determinar la siguiente arista en el contorno.

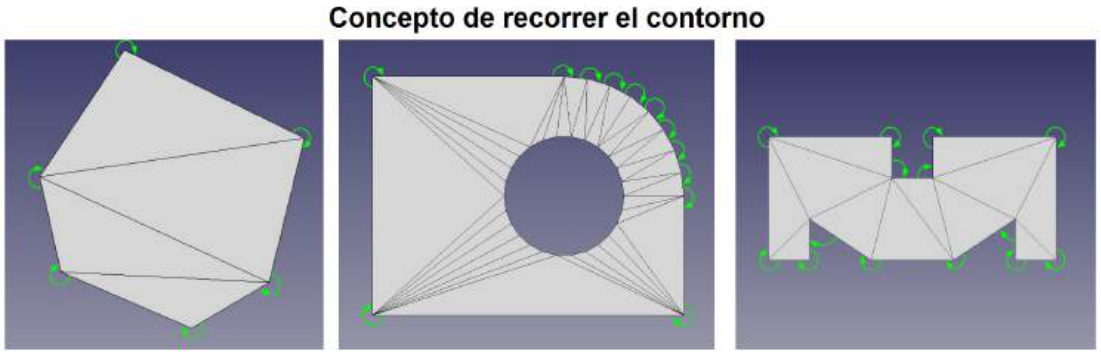


Figura 9.15. Ilustración del concepto de recorrer el contorno.

En la figura 9.16 se ilustra un caso que se considera interesante en el problema de identificar el contorno. Cuando el vértice que se está analizando está “en medio” o “rodeado” de los vértices a los que está conectado. En este caso, algún otro método de análisis podría generar conflicto. Por ejemplo, comparar el valor de los vértices es decir, identificar qué vértice está a la izquierda si tiene un valor menor en el eje x, o a la derecha si tiene un valor mayor. O en el caso del eje Y encontrar si está arriba o abajo. Como se observa el vértice de análisis está en verde, y está rodeado de los vertieces a los que está conectado, que se muestran en amarillo. En verde también se muestra la arista “actual” o de referencia para medir el ángulo. En naranja se muestra la arista “siguiente en el contorno”. Se observa que de los ángulos que se forman entre todas las aristas con la arista “actual” el ángulo formado con la arista “naranja” es el menor.

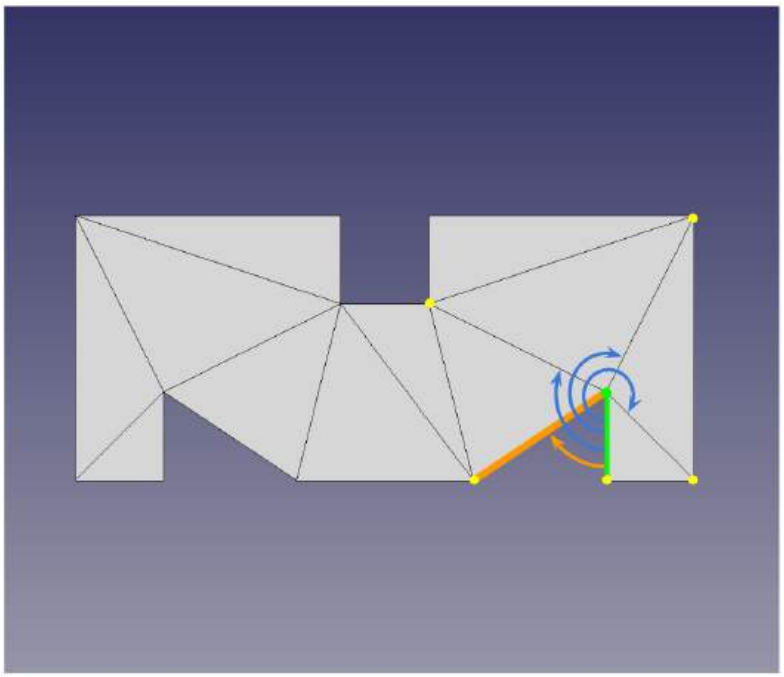


Figura 9.16. Vértice de análisis, rodeado por vértices conectados.

Recordemos que el formato STL solo es una lista de triángulos, con 3 vértices cada uno, por lo que los vértices se repiten tantas veces como se usen. Por lo que para no “excluir” o pasar por alto algún vértice conectado es necesario revisar todas las apariciones del vértice “actual”. Ya que a pesar que ya se haya analizado un vértice anterior, puede ser el caso que en ese “triángulo”, el contorno pase por todas las aristas. También el orden

de aparición de las aristas en el contorno pueden estar intercalados, por lo que triángulos "anteriores" vuelven a ser parte del contorno en más adelante en el análisis.

9.1.6. Identificación del geometrías individuales

Ahora se desea identificar geometrías individuales. En el caso del método 1 se van a identificar líneas rectas y arcos, pero en el método 2 se reduce a únicamente líneas rectas. Como se muestra en la figura 9.17, llegado a este punto ya se tiene identificada la superficie, y también se identificó el contorno. Sin embargo el contorno son solo "fragmentos" o aristas aisladas las cuales no están clasificadas o se pueden considerar un tipo de geometría. Por lo que se desean identificar como se muestra en la imagen en verde las líneas rectas y en morado los arcos.

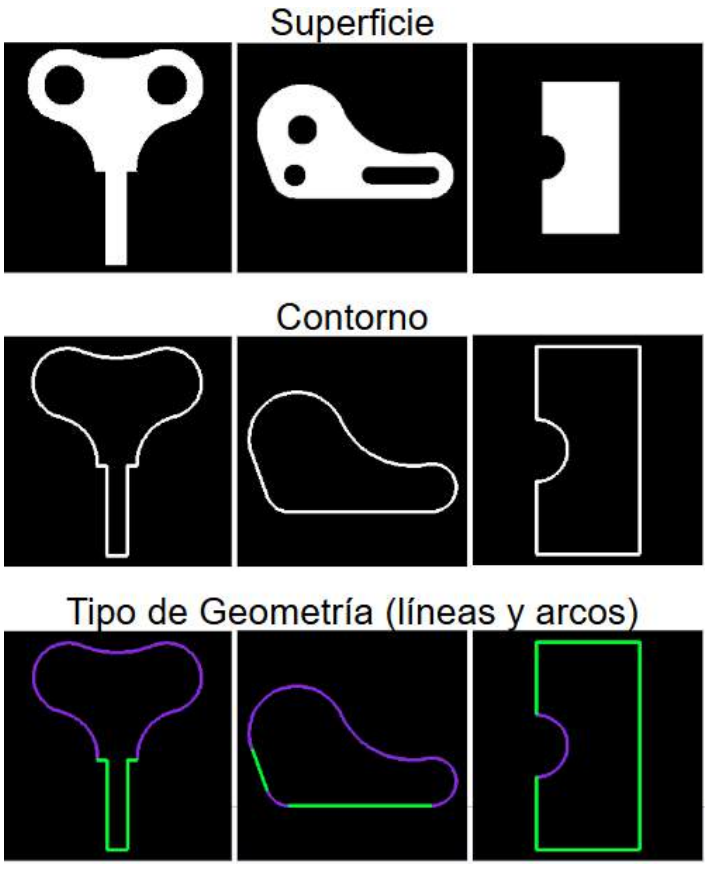


Figura 9.17. Clasificación de las geometrías.

En la figura 9.18 se muestra un caso donde están resaltadas las artistas del contorno pero de nueva cuenta cada arista es un elemento individual. No se tiene información que

ese “conjunto” de aristas forme parte de una misma geometría en este caso de un arco. Para ilustrar las aristas individuales se muestra en color azul y amarillo de forma alternada cada una de ellas.

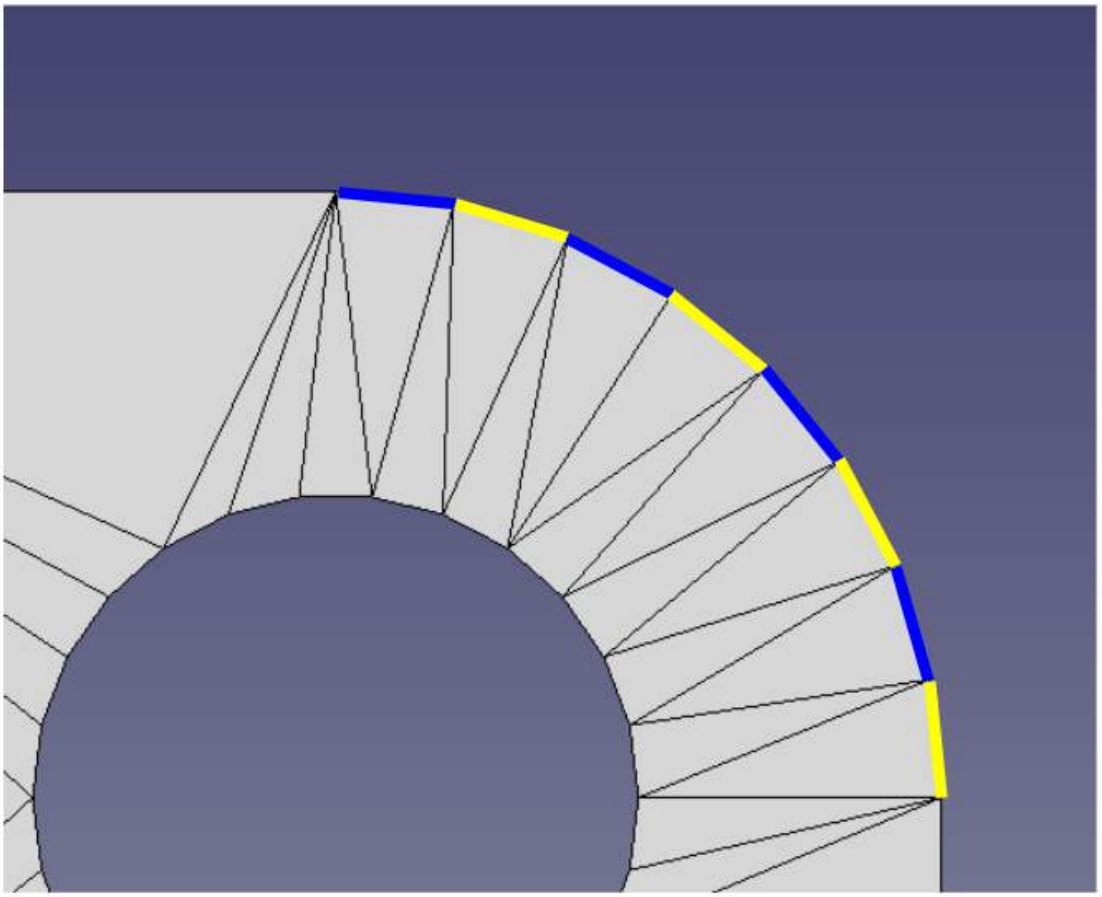


Figura 9.18. Aristas Individuales en un arco.

Con el contorno identificado el siguiente paso es reconocer las “geometrías primitivas” en este caso solo hay dos tipos de geometrías primitivas que son: líneas y arcos.

Se va a iniciar con la explicación para reconocer los arcos. Se realizan varias verificaciones, unas de ellas es el caso que el contorno que estemos analizando sea un polígono regular o un círculo. En estos caso nuestro planteamiento para identificar arcos tendrá problemas, por lo que se revisa si las longitudes de las líneas son iguales con una pequeña tolerancia, y si los ángulos formados entre las líneas son igual también con una ligera tolerancia, en este caso estaremos trabajando con un contorno regular o con un círculo.

Para discriminar entre un polígono regular y un círculo, se agregó un “límite arbitrario” en el cual si un contorno tiene más de 27 lados ya se considera como un círculo. Este número surge de pruebas empíricas usando círculos pequeños de 1 mm de diámetro en FreeCAD para identificar cuántos vértices usaría el formato STL para describirlos. Es importante notar que esta es una simplificación arbitraria hasta cierto punto.

Ahora para la parte de la identificación de los arcos se define que un arco mínimo debe tener 5 vértices. El primer paso para identificar el arco es definir un arco “inicial”, para eso se toman los tres primeros vértices, con ellos se define el “arco de análisi”, ahora se revisan que los siguiente dos vértices también formen parte de es “arco de análisis”, para ello se compara que tengan el mismo centro y el mismo radio. Si los primero 5 vértices cumplen este criterio ya se considera un arco, pero al análisis no se detiene ahí, se siguen revisando todos los siguientes vértices del contorno hasta que algún vértice ya no cumpla con los requisitos.

Para el caso de las líneas rectas, se hacen verificaciones en función de qué longitud tienen las líneas y los ángulos que forman entre ellas. Como ya se discutió, en el caso “especial” en que todos los ángulos son iguales y si todas las longitudes son iguales y si el número de vértices en el contorno es menor de 28, se considera como un polígono regular por ejemplo un pentágono. En el caso de un rectángulo por ejemplo tendríamos que todos los ángulos son iguales es decir de 90 grados, pero las longitudes son diferentes.

Se realiza otra verificación para saber si existen “ángulos de arcos” o “redondeos” esto se realiza al verificar si el ángulo formado entre las “líneas” es mayor a 15 grados o menor a 345 para arcos que se formen en ambas direcciones, en este caso también si no existen ángulos de “arco” se considera que todas las líneas son “rectas”.

El caso interesante es cuando no todos los ángulos son iguales, ni todas las longitudes son iguales, y también está activada la bandera de que existen ángulos de arco. Aquí para poder determinar si una línea es línea “recta” se recurre a analizar su longitud. En este caso se analizan todas las longitudes de las líneas y se define una tolerancia, de la

longitud mínima se agrega un múltiplo de 3.4 para definir una tolerancia, si la longitud de una línea es mayor a esta tolerancia variable se considera que es una línea recta.

9.2. Reconstrucción del sketch a partir de Imágenes.

Ahora se va a plantear el método 2 en el cual se amplía el alcance para incluir imágenes, estas imágenes pueden provenir de diferentes fuentes, por ejemplo pueden venir de fotografías, de capturas de pantalla, de imagenes de internet, o también del proceso descrito en el método 1 el cual puede generar capturas de la cara "principal" de un archivo STL.

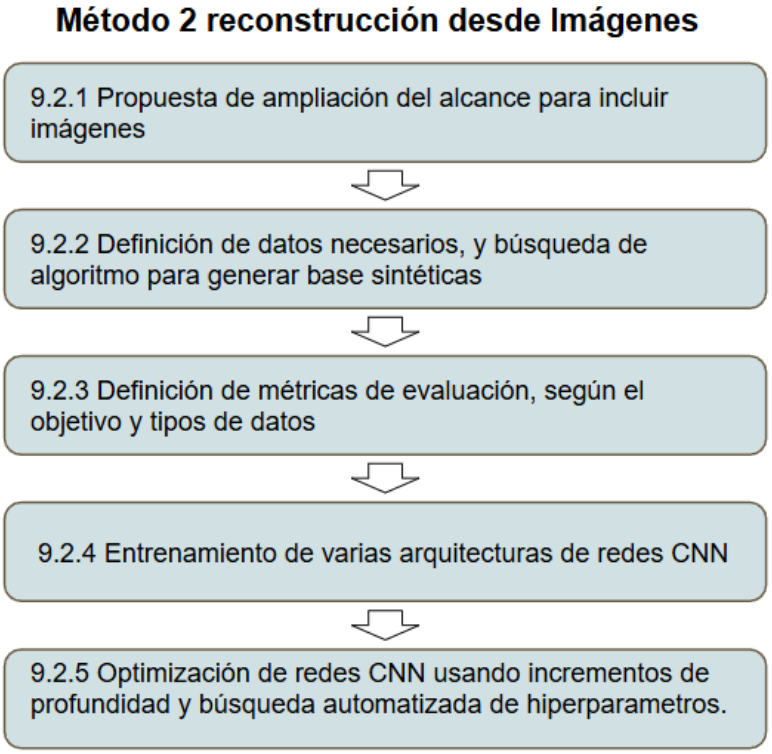


Figura 9.19. Metodología para la reconstrucción del Sketch desde Imágenes.

Se muestra el proceso del método 2 en la figura 9.19. A grandes rasgos la metodología que se siguió fue la siguiente. Primero se plantea ampliar el alcance del proyecto, esto va a permitir procesar más archivos y en un futuro incorporar fotografías. Ya que las redes neuronales convolucionales han demostrado gran capacidad al procesar imágenes

se propone usar este tipo de redes. Para entrenar las redes neuronales se requiere una gran cantidad de datos, por lo que para poder satisfacer esta necesidad se opta por usar datos generados de forma sintética. Se definen el tipo de datos que vamos a generar en función del problema que se quiere resolver así como de limitaciones de cómputo. Se define el tipo de métricas para evaluar las redes neuronales y poder compararlas. Una vez con los datos generados se probaron varios tipos de arquitecturas de redes CNN. Primero se modificaron los hiperparámetros de forma empírica, posteriormente se usó la técnica de ampliar la profundidad de la red, y por último se utilizó una herramientas para buscar hiperparametros de forma automatizada. Con estas pruebas de redes con diferentes hiperparametros se selecciona la red con el mejor desempeño según las métricas definidas. Los datos que es capaz de extraer la red CNN se usan para reconstruir el sketch en el software de diseño mecánico FreeCAD.

9.2.1. Ampliación del alcance

El método 1 de reconstrucción del sketch a partir archivos STL tiene como principal defecto la restricción de solo utilizar ese tipo de archivo de entrada, en búsqueda de ampliar el avance del proyecto y con esto su utilidad se buscó ampliar el tipo de archivos de entrada. Se plantea usar imágenes, ya que las imágenes se pueden generar de diversas fuentes, así como ser capturas de objetos reales mediante fotografías.

El nuevo método planteado llamado “método 2” reconstrucción desde imágenes, es capaz de analizar los siguiente tipo de entradas: capturas de pantalla, imágenes generadas sintéticamente, archivos STL, entre otras. El análisis de archivos STL se logra al combinar partes del método 1 y generar imágenes para ingresar al método 2. Ya que el algoritmo encargado de procesar los archivos STL es capaz de: identificar la superficie principal, reorientar el objeto, identificar el contorno, y generar una imagen del contorno en blanco con fondo en negro. Estas características hacen que ambos métodos sean compatibles dotando al “método 2” de la capacidad de analizar archivos STL también.

Una de las principales razones de usar imágenes, es poder en un trabajo futuro tomar como archivos de entradas fotografías tomadas a objetos “reales” las cuales puedan ser analizadas para reconstruir el sketch de forma automática. En el alcance de este trabajo

este proceso quedó fuera para acotar los límites. Pero las características buscan hacerlo compatible con fotografías en algún punto. Las fotografías son un método muy accesible para la digitalización de un objeto, por ello amplían el alcance del proyecto.

9.2.2. Base de datos sintética. Definición de datos a generar y pruebas de algoritmos de generación

Como se mencionó se van a usar datos generados de forma sintética por la gran cantidad de datos que requiere una CNN para entrenarse. A conocimiento de los autores de este trabajo no existe una base de datos con las características requeridas.

Antes de generar los datos se requiere definir el tipo de datos que se necesita, para resolver el problema deseado. En este caso lo que se desea es extraer las "coordenadas" de los vértices de imágenes de contornos.

Para simplificar el análisis se especifican las siguiente limitaciones. Se requieren imágenes pequeñas de 128x128 píxeles. Las imagenes solo incluyen el contorno en blanco y tienen fondo negro. Los contornos consisten únicamente de líneas rectas, esto con el fin de simplificar la generación de la base de datos así como del análisis. Los contornos pueden tener un máximo de 6 líneas rectas, esto se agrega para generar un tamaño de resolución de la red CNN de tamaño constante.

Para generar la base de datos se propone el siguiente algoritmo. Lo primero es generar un número aleatorio entre 3 y 6 para saber cuántas aristas van a tener el contorno que se va a generar con este límite se generan "zonas" donde puede aparecer un vértice. En la figura 9.20 se muestra un ejemplo para un contorno con 4 vértices. Donde tenemos un área de 128 por 128 píxeles, con los tres límites marcados en rojo.

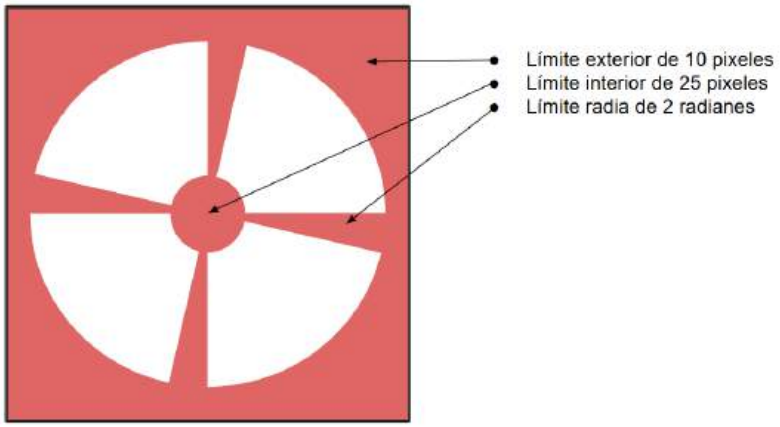


Figura 9.20. Límites para la generación de base de datos.

Ahora se generan coordenadas polares dentro de los límites previamente establecidos. En la figura 9.21 se muestra un ejemplo con 4 vértices en verde que se generan en las áreas establecidas. Estos vértices a sus vez se convierten a coordenadas polares en pixeles, y serán la salida, o respuesta deseada en nuestra CNN, también se puede entender como la "etiqueta".

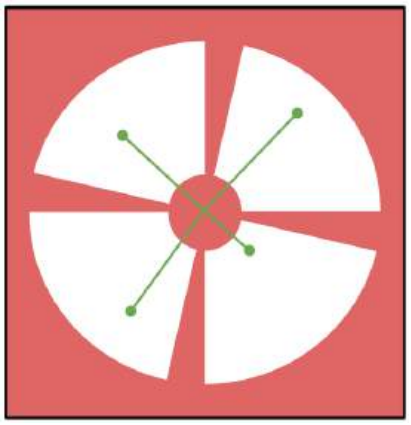


Figura 9.21. Vértices generados para la figura sintética.

En este punto ya tenemos los datos de salida o etiquetas pero necesitamos el datos de entrada que será la imagen por lo que se genera una imagen usando los vértices anteriores. Se muestra un ejemplo en la figura 9.22, donde los vértices se unen para formar el "contorno" el cual se muestra en azul.

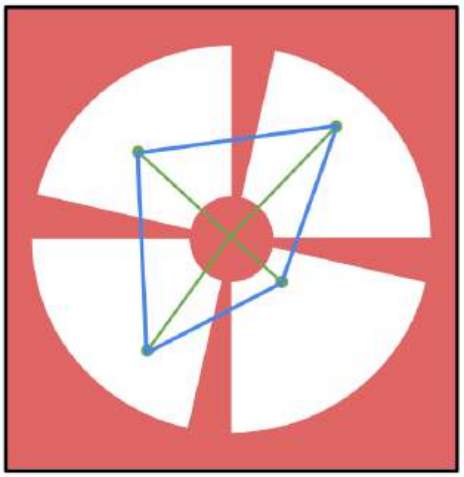


Figura 9.22. Contorno generado en azul.

Ahora para ilustrar cómo se ve el resultado de este proceso en la realidad tenemos la figura 9.23, donde se aprecia el contorno generado de forma sintética.

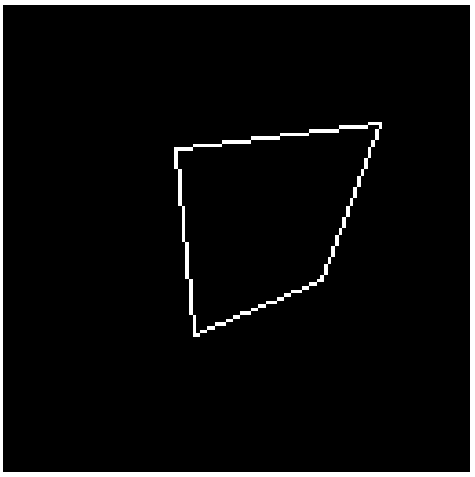


Figura 9.23. Contorno generado de forma automática. Ejemplo específico de 4 líneas rectas de la figura archivo_16.

Las coordenadas de esta figura son la siguientes, como nota por la forma de generar las imágenes el eje Y está invertido es decir inicia en la esquina superior izquierda:

x1 = 87 y1=75

x2 = 52 y2=90

x3 = 47 y3=39

x4 = 103 y4=32

9.2.3. Definición de las métricas de evaluación

Para las métricas de evaluación, se consideran dos aspectos, los datos de salida que se van a generar son datos numéricos, pero a su vez estos datos representan una figura geométrica. Se podrían usar errores como el error cuadrado medio, pero este distorsiona ligeramente los valores y como es importante analizar geométricamente cuando se desvió en cada eje se opta por usar el error absoluto medio, es cual solo tiene en consideración la variación sin tomar en cuenta el sentido.

Por otro lado, para evaluar la figura reconstruida con los valores de los vértices extraídos. Se propone usar una métrica enfocada a la visión por computadora en este caso se usa proporción de Intersección sobre unión la cual nos dice que tanto nuestra figura de predicción se sobrepone con la figura original y a su vez al considerar la union tambien toma en cuenta las porciones que se salen o los “errores”. De esta forma tenemos que cuando hay una alineación perfecta tendremos una IoU de 1.

Ya que se van a analizar varios datos, se considera importante incluir una métrica que analice cómo se comportan los datos en conjunto por lo que se agrega la desviación estándar para saber que CNN tiene mejor desempeño en todos los datos.

Las fórmulas de estas métricas se describen en el apartado de marco teórico, y aquí se aborda el proceso de selección de métricas.

9.2.4. Entrenamiento de redes CNN

Se propusieron y probaron diferentes arquitecturas de redes convolucionales, no son arquitecturas anteriormente publicadas o disponibles en la literatura. Los hiperparámetros que se modificaron fueron: la cantidad de capas convolucionales, el tamaño de los filtros o kernels, la cantidad de capas de submuestreo, la cantidad de capas densas, el tamaño de las capas densas, y por último la función de pérdida.

Tabla 9.1. Arquitecturas usadas.

Versión	16	17	18	19	21	22	op_2
Capas	c1_2	c1_2	c1_2	c1_6	c1_2	c1_2	c1_16
	p1	p1	p1	p1	c2_4	c2_4	p1
	c2_4	c2_4	c2_4,	c2_10	p1	p1	c2_256
	p2	p2	p2	p2	c3_8	c3_8	p2
	c3_8	c3_8	c3_8	c3_10	c4_16	c4_16	c3_256
	p3	p3	p3	p3	p2	p2	p3
	d1_512	d1_512	d1_512	d1_512	c5_32	c5_32	c4_4
	d2_128	d2_128	d2_128	d2_128	c6_64	c6_64	p4
	d3_12	d3_12	d3_12	d3_12	p3	p3	c5_32
					d1_512	d1_512	p5
					d2_128	d2_128	d1_512
					d3_12	d3_12	d2_12

Para comprobar las arquitecturas y ver cómo cambian su estructura se propuso la siguiente nomenclatura. Las capas de convoluciones se nombraron con una letra c, seguida por el número de capa o de convolución que es, luego un guion bajo y por último el número de filtros o mapas de características de esa capa convolucional. Por ejemplo en la arquitectura 17 tenemos “c3_128” esto significa que es la capa convolucional número 3, y que tiene 128 mapas de características o filtros.

Para las capas de “submuestreo”, se usó la letra p, seguida por el número de capa de “submuestreo” que es. Todas son de tipo “submuestreo promedio” de tamaño 2x2 y de paso o “stirde” de 1.

De forma similar las capas densas o completamente conectas se describen con la letra “d” seguida del número de capa densa que es, luego separado por un guion bajo el número de neuronas. Todas las capas densas usan función de activación ReLU.

Los siguientes hiperparámetros son iguales en todas las arquitecturas, el tamaño de kernel (3x3), “stride” o paso de 1, y la función de activación (ReLU). La descripción de

las arquitecturas se muestra en la tabla 9.1, donde también se puede observar que al final de cada red siempre tenemos conectada una capa densa de 12 neuronas. La razón de estas 12 neuronas es para que representen las 6 coordenadas en el plano con cada una de sus dos componentes en x,y.

Los hiperparametros usados se muestran en la tabla 9.2. Algunas arquitecturas son iguales pero con diferentes hiperparametros estas se nombran como una versión independiente para diferenciarlas a pesar de la similitud.

Tabla 9.2. Hiperparámetros usados en las arquitecturas propuestas.

Ver	Función Pérdida	Épocas	Lote	Capa Conv	Filtros	Opt.	Capa Densa	Tamaño Capas Densas
16	MAE	30	128	3	2, 4, 8	Adam	3	512, 128, 12
17	MAE	150	128	3	2, 4, 8	Adam	3	512, 128, 12
18	MSLE	150	128	3	2, 4, 8	Adam	3	512, 128, 12
19	MAE	100	128	3	6, 10, 10	Adam	3	512, 128, 12
21	MAE	150	128	6	2, 4, 6, 8, 16, 32	Adam	3	512, 128, 12
22	MSE	150	128	6	2, 4, 6, 8, 16, 32	Adam	3	512, 128, 12
op_2	MSE	150	32	5	16, 256, 256, 4, 32	Adam	2	512, 12

9.2.5. Optimización de redes CNN

Se usaron 3 métodos para modificar los hiperparámetros de las arquitecturas. El primer método fue empírico, se variaron los hiperparametros y se observó el cambio; esto se aplica a las arquitectura 16, 17, 18 y 19 . El segundo método fue ampliar la profundidad [22]; aplicado a las arquitecturas 21 y 22. El tercer método fue usar la librería optuna para automatizar la búsqueda de hiperparámetros, de está búsqueda automatizada surge la arquitectura versión op_2.

Probar de forma manual diferentes combinaciones de arquitecturas, consume mucho tiempo por lo que poder automatizar las pruebas y variaciones de redes es muy útil. Para eso se usó un método automatizado de optimización, usando la librería de optuna la cual fue publicada en [23], usando esta librería se exploró entre 1 y 5 capas convolucionales, variando los filtros desde 2 hasta 256 subiendo en potencias de 2, se mantiene constante el tamaño del kernel 3x3, la función de activación ReLU, y después de cada capa convolucional, se aplica "submuestreo promedio" de 2x2. También se varía entre 1 y 3 capas densas, de entre 64 a 512 neuronas que aumenta en potencias de 2.

Resultado de la búsqueda de hiperparámetros se obtiene la arquitectura op_2. Es la mejor arquitectura que se pudo obtener en una corrida variando y combinando "algunos" de los hiperparámetros. No necesariamente es la arquitectura óptima variando esos hiperparámetros, ya que probar todas las posibles combinaciones es muy extenso y excede al alcance de este trabajo.

Como se comentará más adelante la versión con el mejor desempeño fue la "17". A grandes rasgos consta de 3 capas convolucionales de 2, 4, y 8 filtros respectivamente. Con capas de submuestreo promedio entre cada convolución. Posteriormente dos capas densas de 512 y 128 respectivamente. Finalmente la capa de salida densa de 12 neuronas (Para 6 líneas por sus 2 componentes x,y). El tamaño de los filtros en las convoluciones es de 3x3, con unidad rectificadora lineal, con paso de 1. En el caso de las capas de submuestreo, son de tipo promedio, con tamaño de 2x2, y paso de 1.

10. Resultados

10.1. Método 1 Reconstrucción desde STL

La base de datos recopilada fue de 102 archivos STL, esto considerando los filtros como caras planas y contornos "sencillos", como se menciona en la metodología. Estos archivos parten de fuentes como repositorios de internet y archivos STL de diseño propio. En la figura 10.1 se muestran algunos ejemplos de los archivos recopilados.

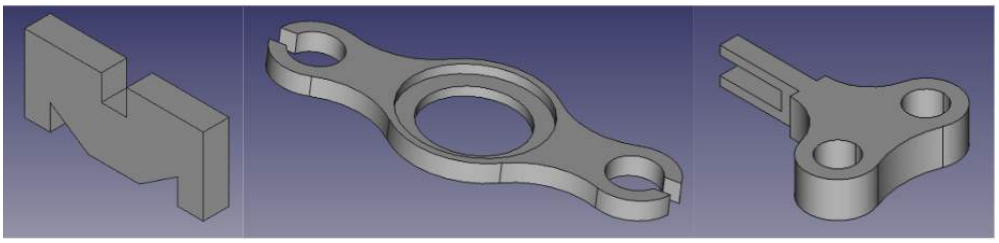


Figura 10.1. Archivos STL de la base de datos.

Pasando a la parte de identificación de la cara principal y del contorno, podemos ver en la figura 10.2 ejemplos de cómo se pudo obtener la cara principal de todas las figuras. También se puede observar que el proceso no solo reorienta la figura en su cara principal, sino que también centra la figura y la escala para que abarque los más posible de imagen.

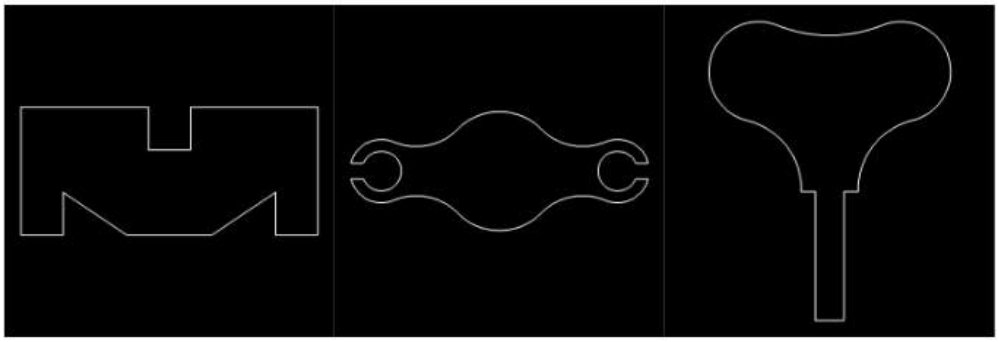


Figura 10.2. Contorno identificado.

A modo de ilustración en la figura 10.3 se muestra un caso de la generación del contorno donde la imagen no estaba centrada ni escalada. El efecto fue corregido.

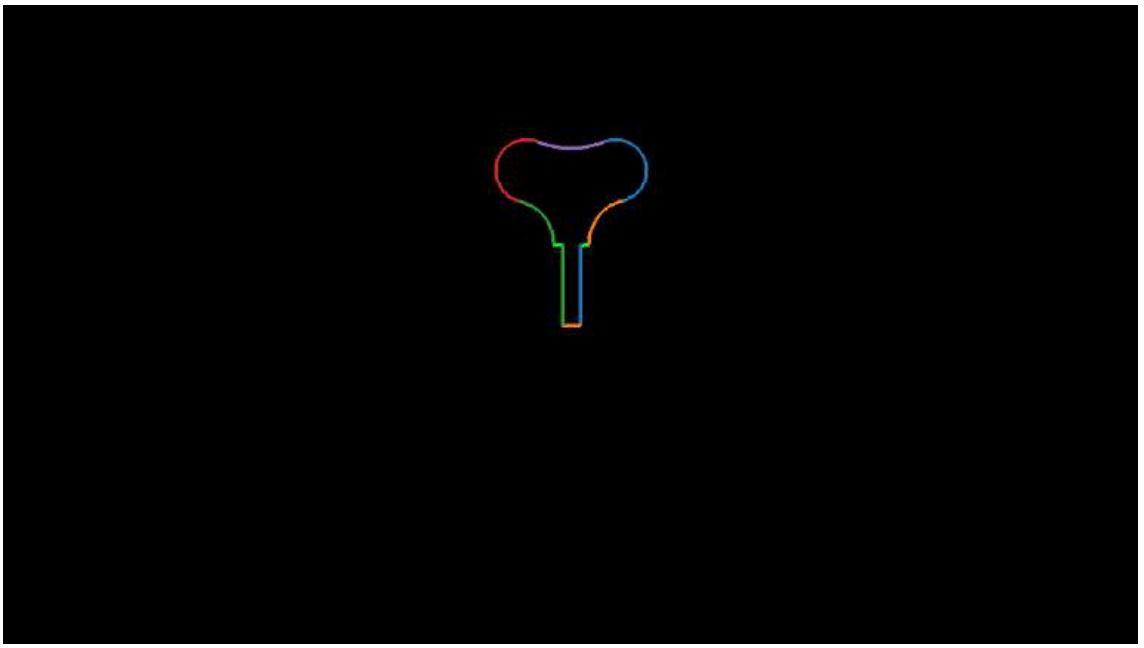


Figura 10.3. Ejemplo de falta de escalado y centrado.

En la figura 10.4 se muestran los tipos de geometrías identificados resaltados en verdes las líneas rectas y en morado los arcos.

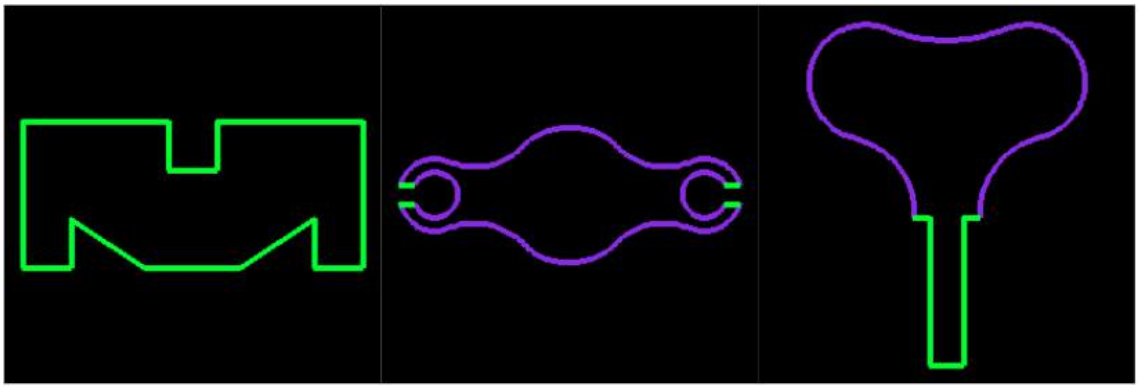


Figura 10.4. Contorno clasificado por tipo de geometría.

En la figura 10.5 se muestran los parámetros o coordenadas extraídos de cada geometría, es decir de cada línea y de cada arco. La figura cuenta con 10 geometrías, 5 líneas rectas y 5 arcos, las dimensiones en este caso están determinadas por el archivos STL. Cada línea recta tiene 4 parámetros, las dos componentes de su punto de inicio y las dos componentes de punto final. A la geometría de tipo arco se le agrega los

parámetros de la coordenada del punto medio. En el caso de las líneas rectas se coloca un “no número” en las coordenadas del centro.

	tipo	coord_start_x	coord_start_y	coord_end_x	coord_end_y	coord_mid_x	coord_mid_x
0	1	337.661065	31.074610	384.382682	177.763632	439.772509	79.336392
1	1	384.382682	177.763632	300.096975	287.963545	322.731180	217.942523
2	0	300.096975	287.963545	278.048488	287.963612	NaN	NaN
3	0	278.048488	287.963612	278.048488	486.400000	NaN	NaN
4	0	278.048488	486.400000	233.951512	486.400000	NaN	NaN
5	0	233.951512	486.400000	233.951512	287.963545	NaN	NaN
6	0	233.951512	287.963545	211.903025	287.963545	NaN	NaN
7	1	211.903025	287.963545	127.617318	177.763632	189.268820	217.942523
8	1	127.617318	177.763632	174.338935	31.074610	72.227491	79.336392
9	1	174.338935	31.074610	337.661065	31.074610	256.000000	46.754551

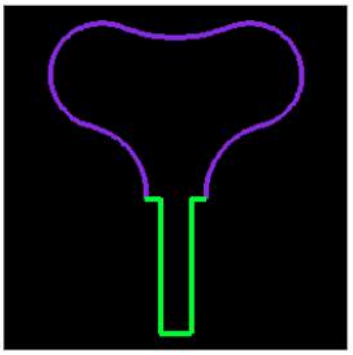


Figura 10.5. Parámetros extraídos del STL.

Ahora con estos parámetros se pueden generar “líneas de código” que describen a cada una de las geometrías. Esto se muestra en el script de la figura 10.6. Al inicio del planteamiento del proyecto se había planteado usar un “modelo grande del lenguaje” para generar el script a partir de los parámetros extraídos, pero se logró simplificar el método. Ya que solo se genera la instrucción a partir del tipo de geometría simplificando el proceso.

```
ScriptsFreecad > 🔷 pureba_ex08_v2.py > ...
1   import FreeCAD as App
2   import Part
3   import Sketcher
4   import FreeCADGui as Gui
5
6   # Crear un nuevo documento
7   doc = App.newDocument("Prisma10x10x30")
8
9   # Crear un nuevo sketch
10  sketch = doc.addObject('Sketcher::SketchObject', 'Sketch')
11  sketch.Placement = App.Placement(App.Vector(0,0,0), App.Rotation(0,0,0,1)) # Puede que nop sea ncesario
12
13  # Definir los puntos del cuadrado
14  sketch.addGeometry(Part.Arc(App.Vector(18.518518447875977, -103.2554702758788, 0), App.Vector(41.67462921142578, -92.31100463867179, 0), App.Vec
15  sketch.addGeometry(Part.Arc(App.Vector(29.11371612548828, -69.99037170410149, 0), App.Vector(15.132824897766113, -60.87888717651361, 0), App.Vec
16  sketch.addGeometry(Part.LineSegment(App.Vector(10.0, -45.00001525878902, 0), App.Vector(5.0, -44.99999999999996, 0)), False)
17  sketch.addGeometry(Part.LineSegment(App.Vector(5.0, -44.99999999999996, 0), App.Vector(5.0, 0.0, 0)), False)
18  sketch.addGeometry(Part.LineSegment(App.Vector(5.0, 0.0, 0), App.Vector(-5.0, 0.0, 0)), False)
19  sketch.addGeometry(Part.LineSegment(App.Vector(-5.0, 0.0, 0), App.Vector(-5.0, -45.00001525878902, 0)), False)
20  sketch.addGeometry(Part.LineSegment(App.Vector(-5.0, -45.00001525878902, 0), App.Vector(-10.0, -45.00001525878902, 0)), False)
21  sketch.addGeometry(Part.Arc(App.Vector(-10.0, -45.00001525878902, 0), App.Vector(-15.132824897766113, -60.87888717651361, 0), App.Vector(-29.113
22  sketch.addGeometry(Part.Arc(App.Vector(-29.11371612548828, -69.99037170410149, 0), App.Vector(-41.67462921142578, -92.31100463867179, 0), App.Ve
23  sketch.addGeometry(Part.Arc(App.Vector(-18.518518447875977, -103.2554702758788, 0), App.Vector(5.506407030103806e-14, -99.6996841430663, 0), App
24
25  # Añadir el sketch al documento y actualizarlo
26  doc.recompute()
27
```

Figura 10.6. Script del sketch.

El script anterior se puede abrir y ejecutar en la GUI de FreeCAD, en la figura 10.7 se muestra una visualización del script en el software de diseño mecánico FreeCAD.

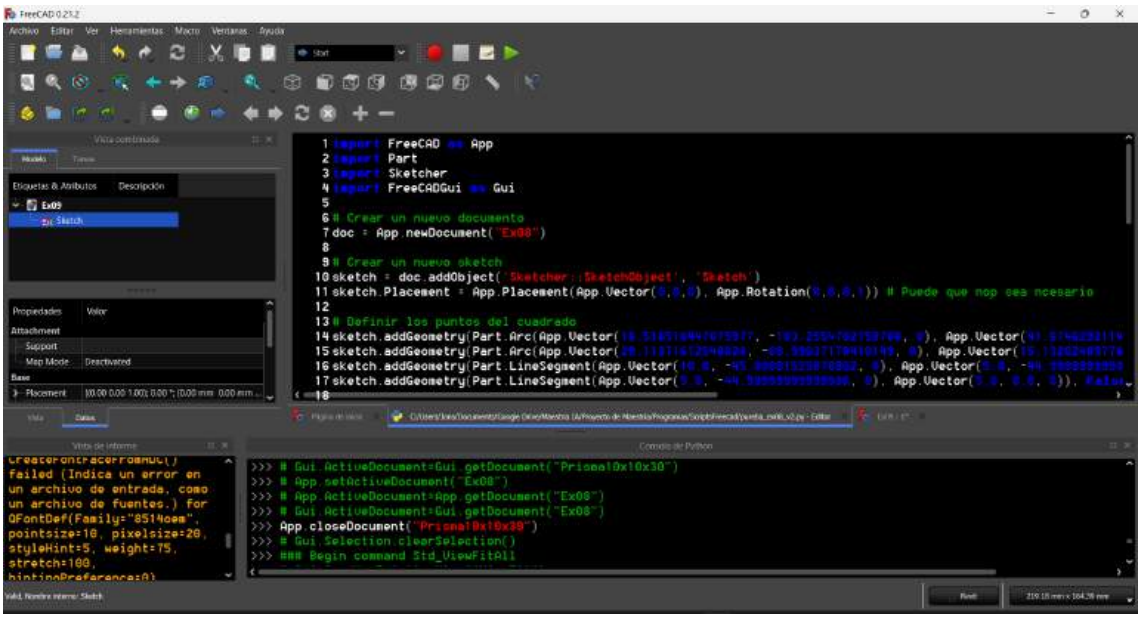


Figura 10.7. Visualización del script del sketch en la interfaz de FreeCAD.

Ahora en la figura 10.8 vemos el resultado de ejecutar este script en el lado derecho vemos en el árbol de operación que a diferencia del previo STL en esta ocasión se ve un objeto de tipo sketch (el símbolo de sketch es diferente no solo el nombre). A diferencia de un STL donde solo se aprecia una "malla" como se menciona anteriormente. También en la parte derecha se observa el contorno o sketch generado en FreeCAD.

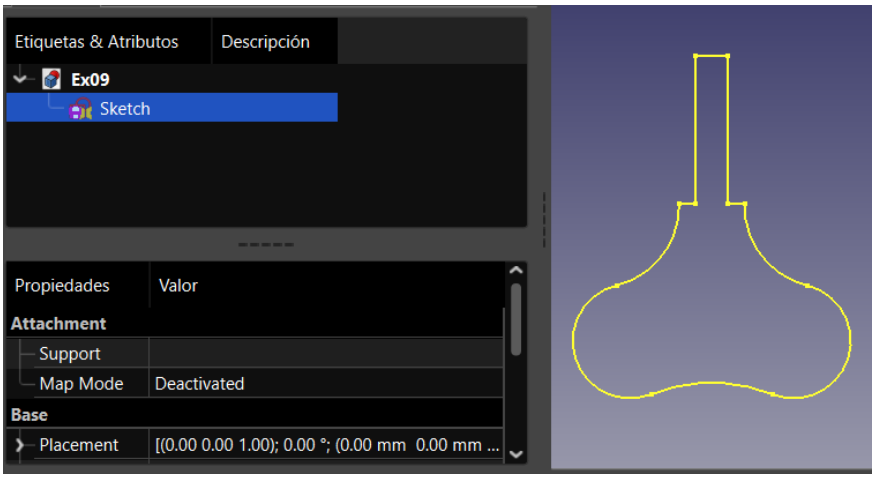


Figura 10.8. Sketch reconstruido en FreeCAD.

Es importante mencionar que con el sketch reconstruido, el archivo es completamente modificable se pueden agregar restricciones geométricas y dimensionales, agregar o modificar geometrías, realizar diversos análisis por ejemplo de análisis de elemento finito, aplicar operaciones volumétricas entre otras opciones.

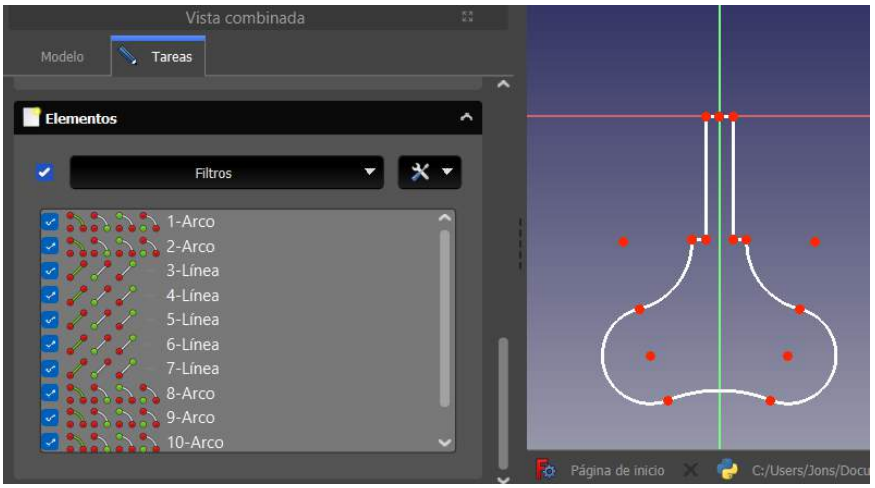


Figura 10.9. Elementos dentro del sketch.

En la figura 10.9 se muestran en la parte izquierda cada uno de los elementos geométricos, ya sean líneas o arcos. Estas geometrías se pueden apreciar al entrar al sketch donde se "enlistan". Se describe el tipo de geometría y el tipo de operación. En este apartado por ejemplo se podría eliminar un elemento.

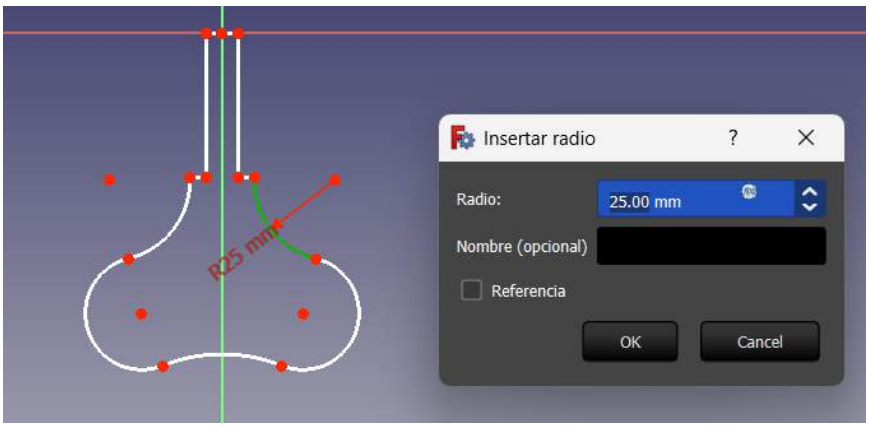


Figura 10.10. Agregar restricción dimensional

En la figura 10.10 vemos como a esos elementos geométricos se le pueden agregar otros parámetros por ejemplo en el caso de la imagen se agrega una restricción de dimensión a uno de los arcos.

10.2. Método 2 Reconstrucción desde imágenes

Al aplicar el método de generación de base de datos sintéticos que se mencionó en el apartado de metodología, se logra generar una base de datos. Los datos e imágenes generadas cuentan con las características deseadas para su implementación. Estas caracteristicas son fondo negro, el contorno en blanco, un tamaño de 128x128, asi como su "etiqueta" que son las coordenadas en X, Y de los vertices. La base de datos final consta de 30,000 pares de imágenes y coordenadas, de entre 3 y 6 líneas o elementos.

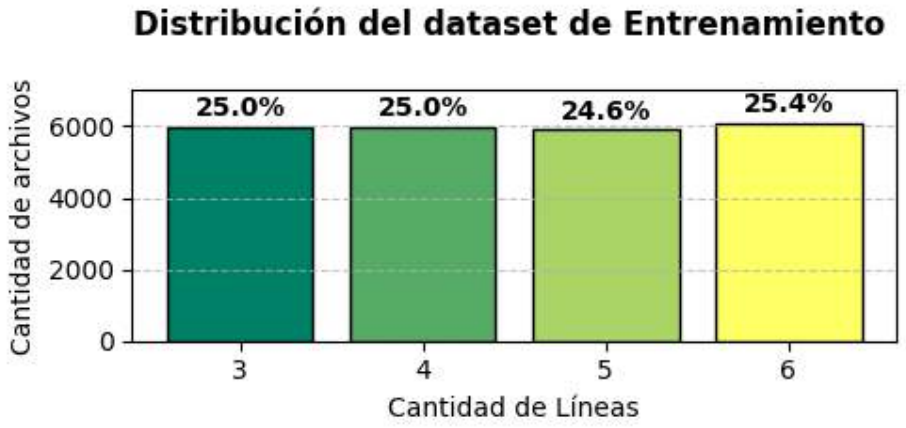


Figura 10.11. Distribución de clases en el dataset de Entrenamiento.

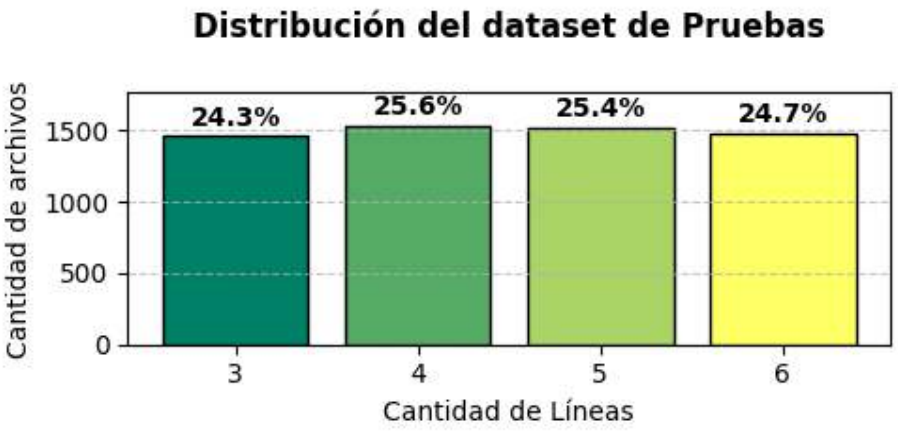


Figura 10.12. Distribución de clases en el dataset de Pruebas.

La base de datos está dividida en 80, 20. Son 24,000 elementos para entrenamiento, y 6,000 para pruebas, respectivamente, se observa la proporción por cada clase o número de líneas. Se tienen 4 "clases" es decir 4 tipos de figuras según su cantidad de "vértices". Se observa en la figura 10.11, y figura 10.12 que cada una de estas clases cuenta aproximadamente con el 25% por lo que están balanceadas, esto aplica para el set de entrenamiento y de pruebas.

La base de datos generada se puede encontrar en el siguiente link de git hub: https://github.com/jonvg7/Dataset-Lines-Contour-Figures. Aquí están en una carpeta la imagenes en formato PNG y en otro carpeta están las coordenadas en pixeles en formato numpy .npy.

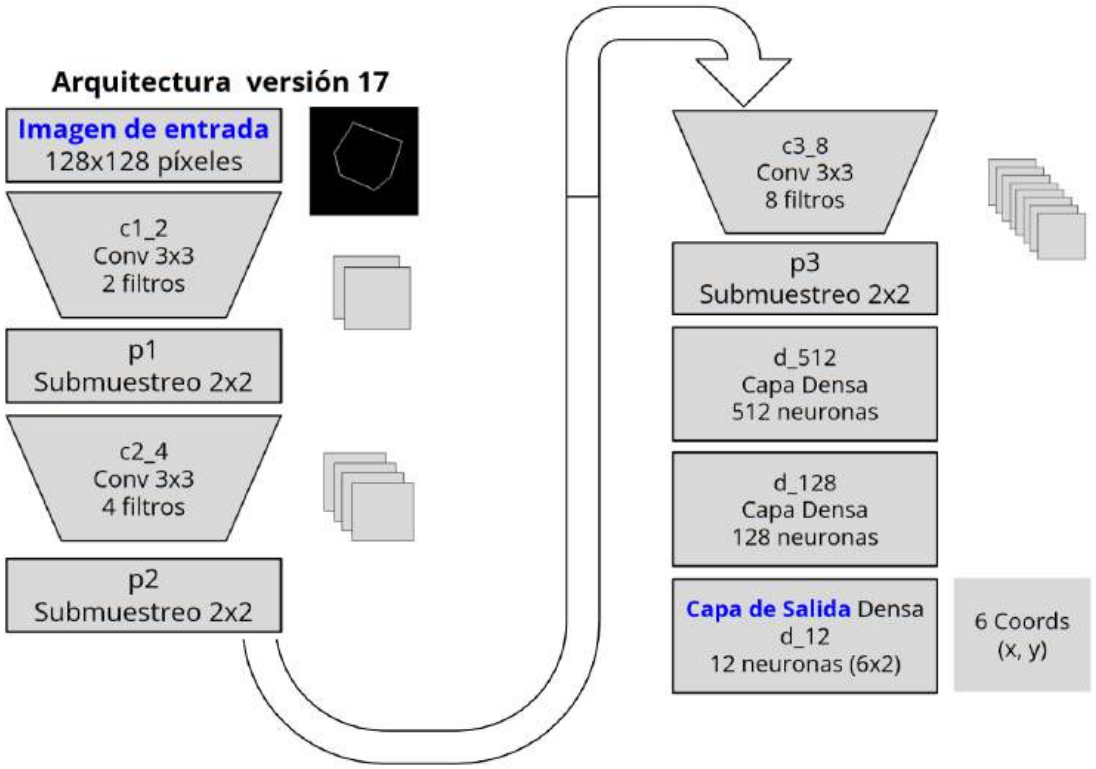


Figura 10.13. Arquitectura usada versión 17.

Después de entrenar las redes propuestas en el apartado de metodología y aplicar las técnicas de optimización, se obtiene un modelo capaz de extraer las coordenadas de los vértices si se le da una imagen como entrada. Esta red se muestra en la figura 10.13 y es la red versión 17. La red genera 12 valores que son la componentes X, Y de hasta 6 posibles vértices. Los valores de las coordenadas van de 0 a 1 pero se convierten a píxeles al multiplicarlos por 128. El orden de los vértices se determina al seguir el orden de las salidas de la red, ya que fueron entrenadas para que las neuronas encuentren los vértices en orden de aparición.

Con estas coordenadas podemos reconstruir el contorno, como se muestra en la figura 10.14, esto es resultado de la red versión 17. En la primera parte se muestran los valores de las coordenadas originales en azul, en la segunda parte se muestra la predicción en verde. En la parte de la derecha se muestra la comparación entre las coordenadas originales y la predicción. Los vértices tanto originales como de predicción se muestran

en rojo.

Para tener mayor información se muestra en la parte inferior, la versión de red que se usó, seguido del índice del archivo, y luego el error MAE y por último el valor de la métrica IoU.

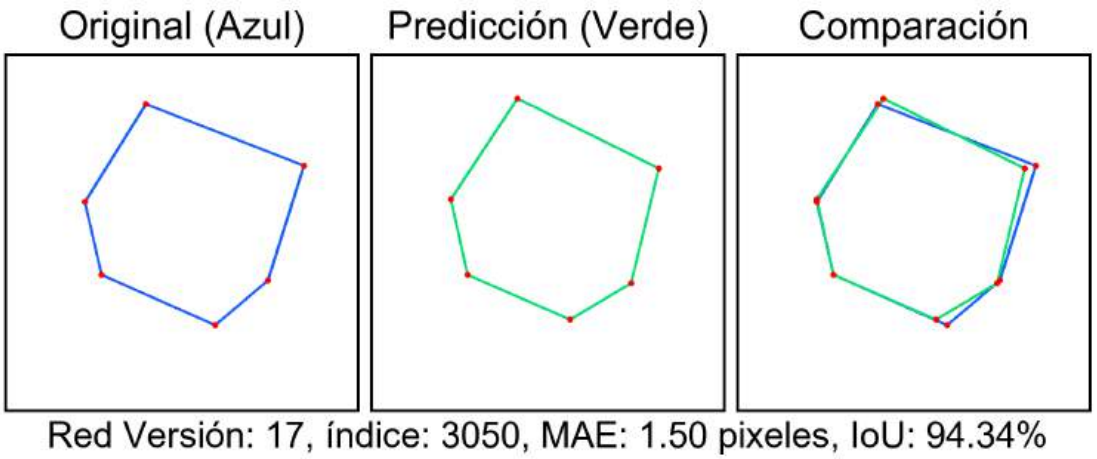


Figura 10.14. Reconstrucción de líneas, red v17 archivo 3050. De izquierda a derecha, la primera en azul son las coordenadas originales, la segunda en verde es la predicción de la Red de las coordenadas, y la última es la comparación entre ambas.

De la comparación entre contorno original y contorno de predicción mostrada en la figura 10.14, se muestra un zoom en la figura 10.15, donde se puede apreciar en rojo los vértices tanto originales como de predicción. En la parte A, se aprecia que la predicción se alinea perfectamente con el original, mientras que en la parte B, se aprecia un desfase de 1 pixel en el eje x, así como 1 pixel en el eje y.

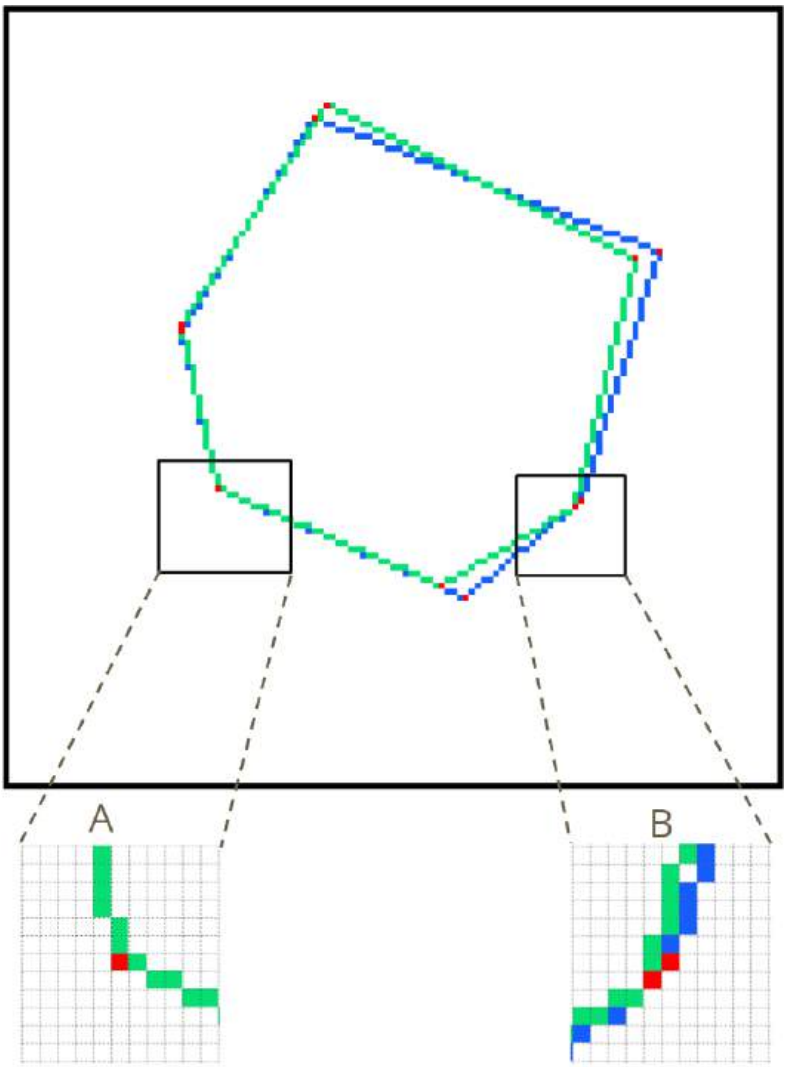


Figura 10.15. Zoom del desfase entre original en azul, y predicción en verde, en rojo se muestra los vértices de ambos contornos.

Para evaluar el rendimiento de las diferentes arquitecturas, se usan las métricas de MAE, IoU, y la desviación estándar DS pero aplicada al MAE para ver que tanto se varía el error MAE a lo largo del grupo de pruebas. Estos resultados se muestran en la tabla 10.1. La tabla está ordenada de mayor a menor según el error MAE. Siendo la arquitectura versión 17 la que tiene mejor desempeño, con el MAE menor y el IoU mayor. El MAE está en pixeles, el IoU es la proporción adimensional entre área de intersección y unión, la desviación estándar DS del error MAE está en pixeles.

Tabla 10.1. Métricas de evaluación de las redes. MAE en pixeles, IoU proporción adimensional, DS en píxeles. Resultados ordenados de peor a mejor del valor MAE.

Ver	MAE	IoU	Error DS	Pérdida Validación
18	4.6625	0.7419	4.7719	0.0363
16	4.0162	0.7699	4.1362	0.0351
22	3.9634	0.7826	4.6866	0.0318
op_2	3.6985	0.7378	3.8119	0.0296
21	2.8938	0.856	4.5813	0.0217
19	2.516	0.8505	3.1407	0.0262
17	***2.3303***	***0.8614***	***3.2693***	***0.0259***

En la figura 10.16 se muestra la distribución del error MAE de la arquitectura versión 17, donde se puede ver, qué porcentaje de los archivos de prueba caen en cierto nivel de error, por ejemplo la primer barra son el 23.5% de los archivos de prueba con un error MAE entre 0 y 1 pixeles, la segunda barra que es el 45.6% tiene un error MAE entre 1 y 2, y así sucesivamente.

Para mostrar cómo se ven cada uno de los niveles de error se van a mostrar imágenes que corresponden a niveles de entre 0 y 1, que es la primera barra, de entre 1 y 2, que es la segunda barra, y de entre 2 y 3. Estas tres barras en conjunto suman el 85% de todas las predicciones realizadas por lo que la mayoría de los datos caen en esta sección.

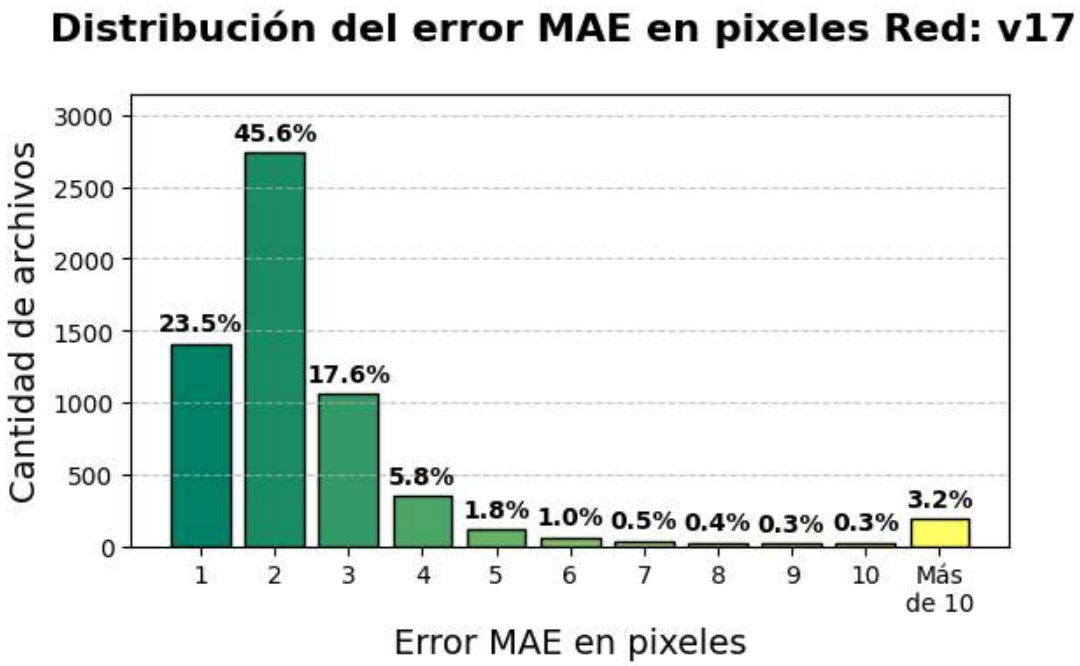


Figura 10.16. Distribución del error MAE, red 17.

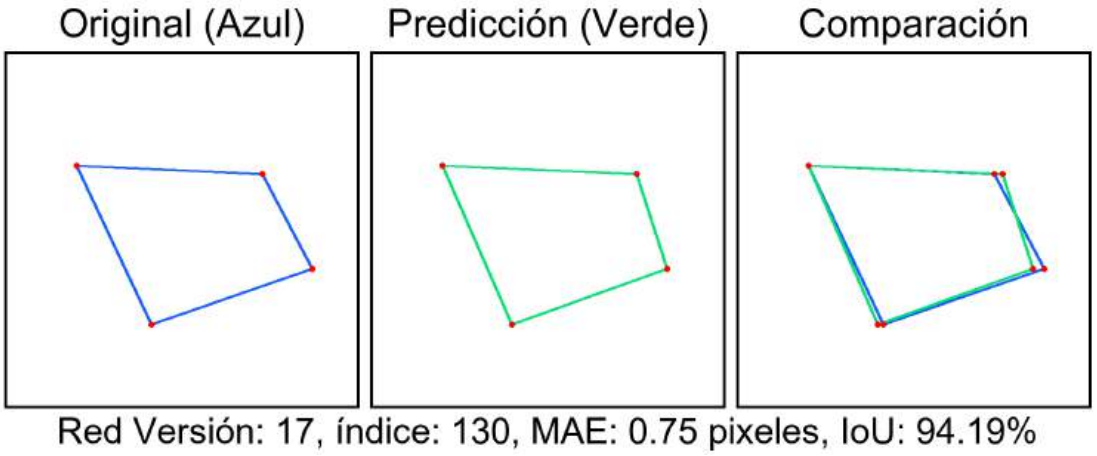


Figura 10.17. Error MAE de entre 0 y 1.

En la figura 10.17 vemos una imagen la cual tiene un error de MAE de 0.75 por lo que iría en la primera barra de la distribución del error, podemos ver como la alineación es casi perfecta. Por ejemplo en la esquina superior izquierda la predicción contra la referencia se alinean perfectamente.

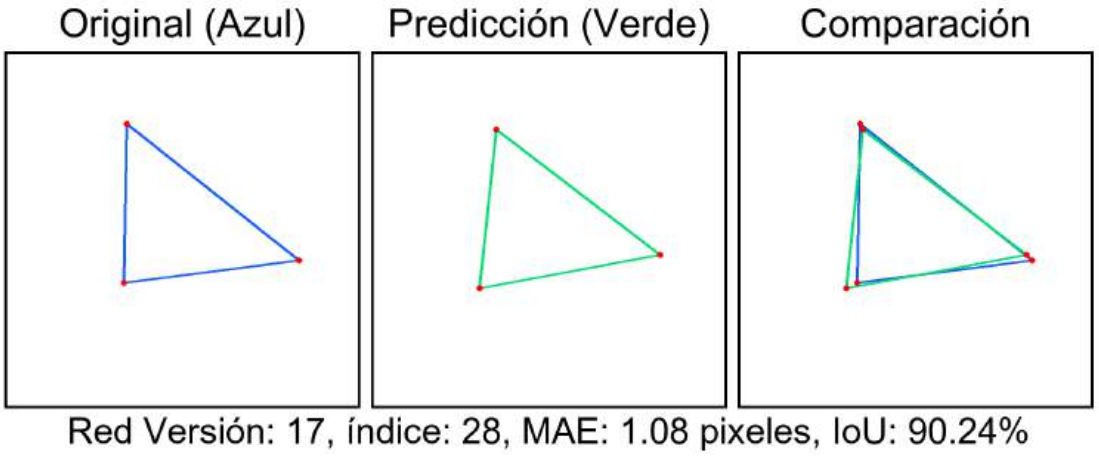


Figura 10.18. Error MAE de entre 1 y 2

En la figura 10.18 vemos un error MAE de 1.08 el cual se sitúa en la segunda barra con valores de entre 1 y 2. Representa el 45.6% de todas las predicciones.

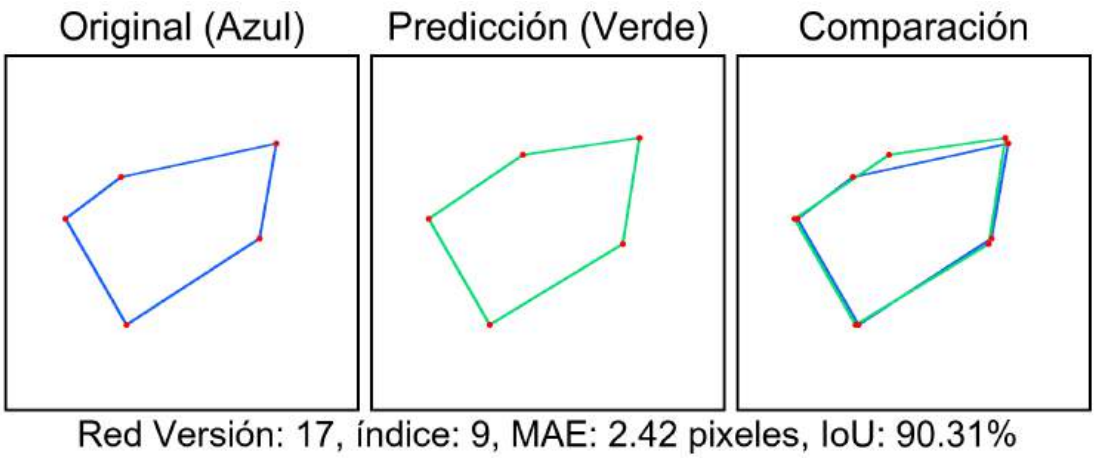


Figura 10.19. Error MAE de entre 2 y 3.

Por último en la figura 10.19 se muestra un MAE de 2.42, lo que corresponde a la barra 3, de entre 2 y 3 de MAE, siendo el 17.6% de los datos.

Se considera de forma arbitraria, que hasta un error MAE de 3 pixeles de error, son buenos resultados. Si se suman los porcentajes de las primeras 3 barras tendremos un porcentaje de acierto "arbitrario" del 86.7%.

Por último con las coordenadas recuperadas se genera el código para describir cada una

de las líneas rectas del contorno con la siguiente instrucción:

sketch.addGeometry(Part.LineSegment(App.Vector(94, -82, 0), App.Vector(72, -95, 0)), False)

```
import FreeCAD as App
import Part
import Sketcher
import FreeCADGui as Gui

# Crear un nuevo documento
doc = App.newDocument("Prueba redes")

# Crear un nuevo sketch
sketch = doc.addObject('Sketcher::SketchObject', 'Sketch')
sketch.Placement = App.Placement(App.Vector(0,0,0), App.Rotation(0,0,0,1))

# Definir los puntos de la figura
sketch.addGeometry(Part.LineSegment(App.Vector(94, -82, 0), App.Vector(72, -95, 0)), False)
sketch.addGeometry(Part.LineSegment(App.Vector(72, -95, 0), App.Vector(35, -79, 0)), False)
sketch.addGeometry(Part.LineSegment(App.Vector(35, -79, 0), App.Vector(29, -52, 0)), False)
sketch.addGeometry(Part.LineSegment(App.Vector(29, -52, 0), App.Vector(53, -16, 0)), False)
sketch.addGeometry(Part.LineSegment(App.Vector(53, -16, 0), App.Vector(104, -41, 0)), False)
sketch.addGeometry(Part.LineSegment(App.Vector(104, -41, 0), App.Vector(94, -82, 0)), False)

# Añadir el sketch al documento y actualizarlo
doc.recompute()
```

Figura 10.20. Script en Python usando la librería de FreeCAD, que se usa para reconstruir el sketch. En otras palabras es el sketch en forma de código.

En la figura 10.20 vemos el script completo, que es una representación del sketch en forma de código. Este código se ejecuta en FreeCAD para poder generar el sketch en la interfaz gráfica de usuario (GUI por sus siglas en inglés), lo cual permite completa manipulación y edición del sketch. Se muestra a partir de la línea de código número 14 cada una de las 6 líneas que componen al contorno, definidas como elementos del sketch. Que en este caso son líneas rectas con sus coordenadas espaciales de inicio y fin.

Por último vemos en la figura 10.21 el resultado de ejecutar el script en la interfaz gráfica de FreeCAD, donde se muestra el sketch completamentamente reconstruido en la sección izquierda de la imagen vemos las 6 líneas que conforman al contorno en las cuales podemos aplicar restricciones geométricas y dimensionales o incluso modificar el tipo de geometría, por ejemplo pasar de una línea a un arco.

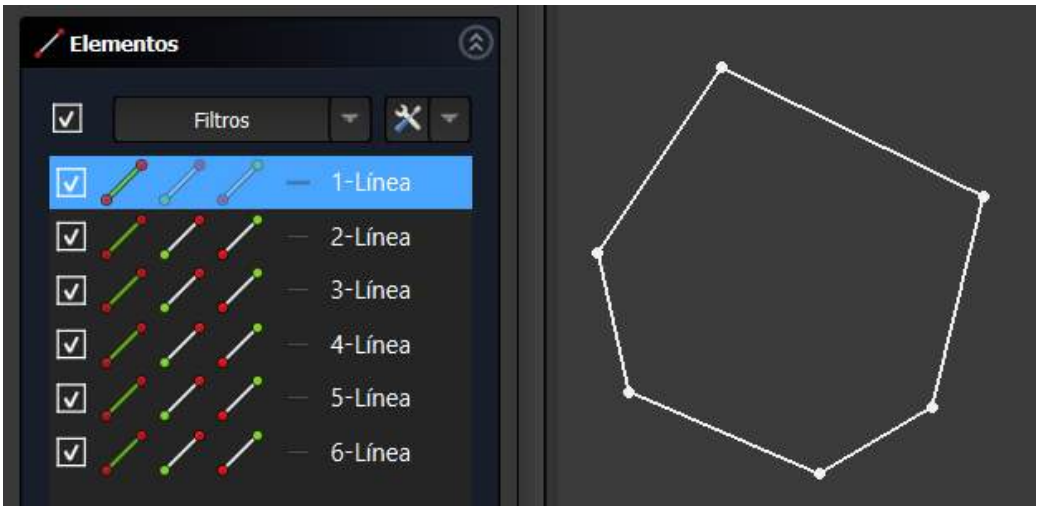


Figura 10.21. Sketch generado en la GUI de FreeCAD. En la parte izquierda se muestran las 6 líneas rectas que componen al sketch. En la parte derecha se muestra la geometría o sketch generada. Todo esto en la GUI de FreeCAD.

11. Conclusión

La reconstrucción del sketch de forma automática abre una gran cantidad de posibilidades, en varias aplicaciones como manufactura, diseño, refacciones, entre otras.

Comparar los resultados con trabajos previos es complicado porque como se menciona en la introducción no se tiene conocimiento de trabajos que sean iguales o que busquen reconstruir el sketch. Sin embargo se pueden realizar comparaciones usando la métrica de IoU por ejemplo los polígonos reconstruidos en [9] logran un IoU del 78%, mientras que los polígonos de [10] logran IoU de entre el 89.9% y 91.3%, por otro lado [11] logra entre 74% y 78% de IoU. Finalmente el enfoque propuesto en este trabajo logra IoU de 86.14% por lo que tiene un desempeño medio siendo superado solo por los resultados de [10], aunque como se menciona los datos de entrada y las salidas no son iguales por lo que estas comparaciones son solo una referencia, y no una comparación directa.

El trabajo actual tiene varias limitaciones restringidas por el tiempo, poder de cómputo

y recursos. El trabajo se puede ampliar, por ejemplo, aquí se trabaja con contornos exclusivamente compuestos por líneas, en un trabajo futuro se puede ampliar las geometrías a arcos y a curvas paramétricas. El tipo de imágenes que se usan son sintéticas, se podrían adquirir imágenes reales, el problema sería encontrar los objetos y las complicaciones asociadas a la adquisición de fotos, como tiempo, almacenamiento, iluminación. Agregar operaciones volumetrías al sketch ayudaría a crear un archivo 3D primitivo.

En un trabajo futuro se podría expandir la optimización de hiperparámetros con optuna, usando un equipo con mayor capacidad de cómputo.

Se considera importante recalcar que los datos usados fueron generados de manera sintética por lo que hay limitaciones como que no tienen similitud con imágenes capturadas de piezas reales o que no tienen deformaciones o ruido de la adquisición. Para simplificar el análisis, se usaron imágenes de máximo 6 líneas rectas, esto limita los escenarios de aplicación del trabajo.

Si bien este trabajo usa datos sintéticos, lo cual limita su aplicación con imágenes de piezas reales, se considera que, la capacidad de reconstruir el sketch, como aquí se muestra, abre una gran cantidad de posibilidades y tiene mucho potencial, esto debido al amplio abanico de aplicaciones que tienes los archivos 3D de diseño mecánico.

Se considera muy importante la capacidad de reconstruir el archivo 3D de una pieza existente, o en el alcance de este trabajo reconstruir el dibujo paramétrico, tener acceso a estos archivos sin la necesidad de software específico, o del conocimiento de diseño pueda abrir la posibilidad a generar refacciones, adaptar piezas existentes, o corregir errores en la pieza que causaron fallas.

Si bien el alcance de este trabajo es limitado ya que no reconstruye el archivo 3D en su totalidad, sería interesante para otro trabajo buscar la forma de agregar operaciones volumétricas al sketch, o también abordar el problema de escanear y generar o analizar imágenes o archivos de mallas de polígonos de piezas ya existentes.

La reconstrucción del sketch y posteriormente del archivo 3D de diseño mecánico, tiene muchas aplicaciones: en manufactura, análisis de ingeniería, diseño de componentes,

fabricación de refacciones, generación de prototipos entre otras. Por lo tanto el presente trabajo tiene mucho potencial en diversas aplicaciones.

12. Referencias

[1] F. Bianconi, “Bridging the gap between cad and cae using stl files,” International Journal of CAD/CAM, vol. 2, pp. 55–67, 01 2002.

[2] K. B. Guo, L. C. Zhang, C. J. Wang, and S. H. Huang, “Boolean operations of stl models based on loop detection,” The International Journal of Advanced Manufacturing Technology, vol. 33, p. 627–633, 06 2007.

[3] H. Park and K. H. Lee, “A new parametric control method for freeform mesh models,” The International Journal of Advanced Manufacturing Technology, vol. 27, pp. 313–320, 2005.

[4] V. Sunil and S. Pande, “Automatic recognition of features from freeform surface cad models,” Computer-Aided Design, vol. 40, no. 4, pp. 502–517, 2008.

[5] W. Para, S. Bhat, P. Guerrero, T. Kelly, N. Mitra, L. J. Guibas, and P. Wonka, “Sketchgen: Generating constrained cad sketches", " Advances in Neural Information Processing Systems, vol. 34, pp. 5077–5088, 2021.

[6] R. Wu, C. Xiao, and C. Zheng, “Deepcad: A deep generative network for computer-aided design models,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6772–6782, 2021.

[7] K. D. Willis, Y. Pu, J. Luo, H. Chu, T. Du, J. G. Lambourne, A. Solar-Lezama, and W. Matusik, “Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences", ACM Transactions on Graphics (TOG), vol. 40, no. 4, pp. 1–24, 2021.

[8] L. Wang and X. Wang, “Ppi-net: End-to-end parametric primitive inference,” in Computer Graphics International Conference. Springer, pp. 67–78, 2023.

[9] Y. Chen, Y. Wu, L. Xu, and A. Wong, “Quantization in Relative Gradient Angle Domain For Building Polygon Estimation,” 2022 26th International Conference on Pattern Recognition (ICPR), Jan. 2021.

[10] S. Zorzi, Shabab Bazrafkan, S. Habenschuss, and Friedrich Fraundorfer, “PolyWorld: Polygonal Building Extraction with Graph Neural Networks in Satellite Images,” arXiv (Cornell University), Nov. 2021.

[11] N. Girard, D. Smirnov, J. Solomon, and Y. Tarabalka, “Polygonal Building Extraction by Frame Field Learning,” Thecvf.com, pp. 5891–5900, 2021.

[12] E. Rosten and T. Drummond, “Machine Learning for High-Speed Corner Detection,” Computer Vision – ECCV 2006, pp. 430–443, 2006.

[13] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016, ch. 9.

[14] G. Fernández-Avilés, J.-M. Montero, Fundamentos de ciencia de datos con R. Madrid, España : McGraw-Hill Interamericana de España S.L., 2024.

[15] S. I. Grossman S. y J. J. Flores Godoy, Álgebra Lineal, 7ma ed. México D.F., México: McGraw-Hill Interamericana, 2012

[16] J. Carrillo, *Geometría analítica*, 2ª ed. Ciudad de México, México: Editorial Patria, 2014.

[17] Roger N. Nagel, Walt W. Braithwaite, Philip R Kennicott, “Initial Graphics Exchange Specification IGES Version 1.0”, National Bureau of Standards, NBSIR 80-1978, 03 1980.

[18] "STEP-file, ISO 10303-21", Library of Congress, 2017.

[19] S. Marjudi, M. F. Mohamad Amran, K. Abdullah, S. Widyarto, N. A. Abd Majid, and R. Sulaiman, A review and comparison of iges and step," 01 2010.

[20] C. Iancu, D. Iancu, and A. St ̆ancioiu, "From cad model to 3d print via stl file format." Fiability & Durability/Fiabilitate si Durabilitate, no. 1, 2010.

[21] M. Szilvási-Nagy and G. Mátyási, "Analysis of stl files" Mathematical and Computer Modelling, vol. 38, no. 7, pp. 945–960, 2003, hungarian Applied Mathematics.

[22] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv.org, 2015.

[23] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Jul. 2019.