UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EVALUACIÓN DE ALGORITMOS DE DETECCIÓN AUTOMÁTICA DE PLANOS DE SIMETRÍA EN NUBES DE PUNTOS 3D

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

GUSTAVO ADOLFO SANTELICES NÚÑEZ

PROFESOR GUÍA:
Ivan Sipiran Mendoza

MIEMBROS DE LA COMISIÓN:
Nancy Hitschfeld Kahler
Daniel Calderón Saavedra

Este trabajo ha sido parcialmente financiado por ANID Fondecyt grant 11220211

SANTIAGO DE CHILE
2024

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN
POR: GUSTAVO ADOLFO SANTELICES NÚÑEZ
FECHA: 2024
PROF. GUÍA: IVAN ANSELMO SIPIRAN MENDOZA

EVALUACIÓN DE ALGORITMOS DE DETECCIÓN AUTOMÁTICA DE PLANOS DE SIMETRÍA EN NUBES DE PUNTOS 3D

La simetría es un fenómeno que se encuentra muy presente en una amplia variedad de contextos. Conocer la simetría de un objeto puede ser útil para varias aplicaciones como la reconstrucción de modelos 3D, a partir de información incompleta o para encontrar anomalías en objetos que son comúnmente simétricos. Esto motiva el desarrollo de algoritmos que detecten de manera automática las simetrías de un objeto.

Dado que la detección automática de simetrías no es una tarea sencilla de resolver, se han desarrollado muchos algoritmos para esta tarea, muchos de los cuales hacen uso de modelos de aprendizaje automático. Sin embargo, comparar los algoritmos desarrollados para este problema es una tarea compleja debido a que todos han sido desarrollados y evaluados con distintos conjuntos de datos, cosa que imposibilita la comparación justa entre algoritmos.

Este informe tiene por objetivo presentar la implementación y evaluación de dos modelos de aprendizaje automático, sobre un dataset especializado para esta tarea. El primer modelo está basado en PRS-Net y utiliza una red convolucional sobre una voxelización de una nube de puntos para predecir una cantidad arbitraria de simetrías de reflexión. Por otra parte, el segundo modelo ocupa una red PointNet directamente sobre la nube de puntos de input para predecir sus simetrías. Ambos modelos utilizan estrategias distintas para predecir las simetrías, por lo que resulta interesante compararlas entre sí.

Los experimentos muestran que el modelo basado en PRS-Net no es capaz de aprender a detectar simetrías de manera automática, mientras que el modelo basado en SymmetryNet obtiene buenos resultados. Se investiga como varían los resultados obtenidos según el tipo de perturbación aplicado y el tipo de figura del input. Se encuentra que el tipo de perturbación no afecta los resultados obtenidos y que existen tipos de figuras para los cuales detectar sus simetrías es más complejo.

A mi familia, este éxito es de todos nosotros.

Agradecimientos

En primer lugar me gustaría agradecer a mis padres por la oportunidad de estudiar y por su apoyo constante. En la dedicatoria digo con mucho orgullo que este éxito es de todos nosotros y es porque así lo siento. Hace un tiempo atrás, cuando tuve la oportunidad de ver una exposición de un proyecto de Ciencia de Datos en educación, uno de los mensajes de los expositores fue que uno de los factores de mayor importancia para poder terminar algún grado escolar con éxito era el apoyo familiar. Hoy puedo decir, con mucha alegría, que mi historia es prueba de ello.

Agradecer en especial a mi hermano menor, José Miguel, quien muchas veces sin hablar dijo todo lo que necesitaba y siempre estuvo ahí para apoyarme y darme ánimo. También agradecer a mi hermano mayor, Pablo, quien siempre me ayuda a ver las cosas desde otra perspectiva y centrarme.

En general, a toda mi familia, son tantos los momentos de apoyo y cariño que me han brindado que podría llenar miles de hojas, gracias por cada uno de ellos. Por último, agradecer a Felipe, mi hermano escogido, por acompañarme en este proceso.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

La simetría es una propiedad geométrica de los objetos, la cual expresa la invariancia de un objeto frente a ciertas transformaciones, como puede ser la reflexión frente a un plano o la rotación en torno a un eje. Por ejemplo, se dice que un objeto de tres dimensiones tiene simetría de reflexión si es que cada punto del objeto tiene una correspondencia, la cual es la reflexión de este punto frente al plano. Es interesante estudiar la simetría debido a su gran cantidad de ocurrencias en la naturaleza y debido a sus propiedades a la hora de resumir la información de una geometría.

Usualmente, los escaneos de estos objetos son obtenidos en forma de nubes de puntos. Esto motiva buscar métodos para encontrar simetrías de forma automática en nubes de puntos. Sin embargo, esto resulta una tarea bastante complicada desde el punto de vista técnico, ya que los métodos utilizados usualmente no son lo suficientemente robustos o solicitan que se cumplan condiciones especiales sobre las nubes de puntos.

El estado del arte se ha enfocado en utilizar redes neuronales para poder desarrollar modelos que permitan identificar simetrías de manera automática. Se han probado arquitecturas ya probadas sobre otras tareas de nubes de puntos, obteniendo resultados prometedores. Sin embargo, actualmente no se cuenta con una metodología de evaluación que permita evaluar el rendimiento de estos modelos y compararlos entre sí.

Es este contexto el que motiva la realización de este trabajo para poder implementar algunos algoritmos de Deep Learning y probarlos en un dataset generado justamente para resolver esta misma tarea. Se espera que el resultado de este trabajo permita tener una primera aproximación de un benchmark donde se presenten modelos conocidos, pero adaptados a esta tarea y dataset.

Algunas aplicaciones que podría tener un método robusto de poder calcular simetrías en nubes de puntos son:

- Reconstrucción de objetos 3D a partir de una imagen de manera no supervisada. [1]
- Análisis y diseño de partes de ingeniería. [2]
- Detección de cáncer mamario. [3]
- Segmentación de órganos. [4]

En particular, en este trabajo se probará con dos arquitecturas de modelos de Deep Learning, PRS-Net [5] y SymmetryNet [6], y se evaluará su rendimiento sobre el dataset presentado en SHREC 2023 [7] creando así este benchmark.

1.1. Objetivo General

El objetivo general del presente informe es evaluar los resultados de algunos métodos del estado del arte de Deep Learning en la tarea de detectar simetrías de reflexión en nubes de puntos sobre un dataset en común para poder compararlos y poder estudiar cuáles modelos obtienen mejores resultados.

Objetivos Específicos

1. Implementar un método de voxelización para poder adaptar el dataset de nubes de puntos a voxels para poder usarlos para entrenar PRS-Net.
2. Adaptar e implementar PRS-Net y entrenarlo sobre el dataset voxelizado.
3. Registrar y evaluar los resultados obtenidos por PRS-Net.
4. Adaptar e implementar la arquitectura de SymmetryNet para que utilice solo nubes de puntos como input.
5. Registrar y evaluar los resultados obtenidos por SymmetryNet.
6. Resumir y ordenar los resultados obtenidos enfocando la reproducibilidad de los experimentos realizados.

1.1.1. Evaluación

Dado que el trabajo se centra en implementar un par de modelos de Deep Learning para poder evaluarlos frente al mismo dataset, lo más relevante de la evaluación son las métricas a utilizarse para evaluar los modelos. Para esto se utilizarán las métricas definidas en la sección 2.5. Con estas métricas se podrá comparar ambos modelos y podrá cumplirse el objetivo al reportar los resultados obtenidos. Es deseable que junto con esto se puedan encontrar algunas explicaciones de estos resultados.

El resto de este informe está organizado de la siguiente manera. En el siguiente capítulo se entrega un marco teórico de los conceptos principales tratados en este informe y se presentan los modelos originales que inspiran los modelos implementados. Luego en el capítulo 3 se describen los modelos adaptados a este dataset y se analizan los resultados obtenidos. Y finalmente, en el capítulo 4 se presentan los aportes principales y los puntos de trabajo futuro.

Capítulo 2

Marco Teórico

2.1. Conceptos previos

2.1.1. Acerca de la simetría

La simetría es un concepto matemático que agrupa un gran conjunto de transformaciones. En [8] se dice que un objeto geométrico M es simétrico con respecto a una transformación T, si $M = T(M)$, es decir, que M es invariante ante la transformación T.

Luego, [8] presenta una taxonomía de los tipos de simetrías según varios criterios. En particular, para el contexto del presente informe, son relevantes 4 criterios. El primer criterio se refiere a si la simetría es global o parcial, esto es, que todo el objeto sea simétrico respecto a una transformación o sólo una parte de él. El segundo, se define según la métrica utilizada para medir la distancia, se dice que una simetría es extrínseca si se utiliza la distancia euclidiana mientras que si se utiliza la distancia geodésica se considera que la simetría es intrínseca. El tercer criterio introduce la noción de las simetrías aproximadas o exactas, donde las aproximadas son aquellas en las que el objeto resultante es muy similar al objeto original mas no es exactamente igual. Y el último criterio se refiere a la clase de transformación aplicada. Para objetos 3D existen 2 clases de transformaciones que dan lugar a simetrías, estas son las reflexiones y las rotaciones.

Por otra parte, también se debe considerar el tipo de objeto geométrico sobre el cual se buscan sus simetrías, por ejemplo, se puede buscar simetrías sobre mallas o nubes de puntos. Esta manera de clasificar hace aparente que existen muchas variaciones del problema de la detección de simetrías según el tipo de simetría que se busca detectar. Estas variaciones motivan el desarrollo de distintas técnicas que se ajusten a las particularidades de cada problema.

En particular, el problema que se busca resolver con los métodos presentados más adelante es el de *detección de simetrías extrínsecas globales aproximadas de reflexión en nubes de puntos 3D*. Este tipo de simetría puede ser caracterizado totalmente median-

te un *plano de simetría* T para el cual si la nube de puntos P presenta una simetría ante T se cumple que P es invariante ante la reflexión sobre T.

2.1.2. Deep Learning con nubes de puntos y PointNet

Las nubes de puntos son un tipo de dato complejo de manejar debido a que son un conjunto desordenado y de densidad variable. Antes de que apareciera PointNet [9], los modelos de Deep learning transformaban las nubes de puntos a representaciones más estructuradas que pudieran procesar. Una alternativa bastante simple, y que intentaba aprovechar las fortalezas de los modelos de imágenes, era pasar las nubes de puntos a representaciones volumétricas como las voxelizaciones. Por ejemplo, un modelo que utilizaba voxelizaciones como representaciones de nubes de puntos es VoxNet [10].

Las principales dificultades que presentan las nubes de puntos radican en que son un conjunto de densidad variable y desordenado. Esto lo soluciona PointNet al utilizar un multi-layer perceptrón (MLP) compartido por cada punto y aplicar una función simétrica para generar el vector de características globales de una nube de puntos. Una función simétrica es aquella que toma una cantidad arbitraria de vectores como entrada y produce un nuevo vector que es invariante al orden de entrada. Por ejemplo, las funciones binarias + y * son funciones simétricas. PointNet al utilizar una función simétrica como función de agregación de los puntos asegura que se puede obtener un vector de características representativo de toda una nube de puntos independiente del orden que tenga. Dado este hecho uno puede usar una red PointNet para entregarle dicho vector de características a otro MLP que se encargue de la tarea en particular que se necesite resolver.

2.1.3. Criterio de igualdad entre planos

En este informe muchas veces se dirá que dos planos son iguales, en particular se tendrá que decidir si un plano predicho es igual a alguno de los reales. Para poder hacer esto se utilizará la siguiente definición:

Definición 1 (Criterio de igualdad entre planos)**.** Dado dos planos $P_1 = [n_1, p_1]$ y $P_2 = [n_2, p_2]$ representados por un punto en el plano y la normal del plano. Se dirá que $P_1 =_{\theta,\epsilon} P_2$ cuando se cumpla que:

$$\text{angle}(n_1, n_2) < \theta \wedge \|p_1 - p_2\| < \epsilon \tag{2.1}$$

En particular, dado 2 conjuntos de planos, uno de planos predichos y otro de planos reales. Se dirá que un plano predicho es verdadero positivo si éste es igual a un plano de los reales y no existe otro plano predicho de mayor confianza que sea igual al plano real con el que se estaba comparando.

2.1.4. Distancia de Chamfer

La distancia de Chamfer [11] es una métrica que mide la similitud entre dos nubes de puntos. Para dos nubes de puntos S_1 y S_2 la distancia de Chamfer (D_{CD}) se define como:

$$D_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2 \tag{2.2}$$

Notar que para una nube de puntos y una simetría, la distancia de Chamfer entre la nube de puntos original y la nube de puntos resultante de aplicar la transformación de simetría es igual 0, ya que son nubes de puntos idénticas.

2.2. PRS-Net

PRS-Net [5] propone utilizar una red convolucional sobre la representación volumétrica de una figura para aprender de manera no supervisada cuáles son las simetrías de reflexión mediante la predicción de los parámetros del plano de simetría asociado y cuáles son las simetrías de rotación según la predicción del eje de rotación. Para esto definen una función de pérdida la cual intenta minimizar el error de distancia de simetría y maximizar la ortogonalidad entre los planos predichos. Finalmente, introducen un proceso para refinar los planos predichos en el que remueven los planos repetidos y aquellos que su error de simetría es muy alto.

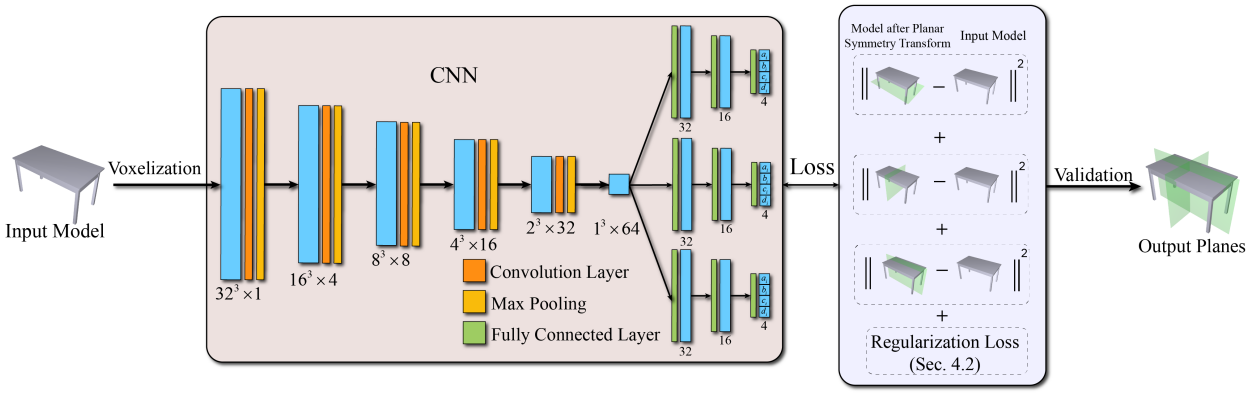

Figura 2.1: Resumen del método descrito por PRS-Net. (Figura extraída de [5])

Al igual que en la fuente original, se presenta el diseño general de la red en la figura 2.1. Esta arquitectura consiste en una red convolucional de 5 capas con kernel de tamaño 3, padding 1 y stride 1. Después de cada capa de convolución se aplica una capa de max pooling con kernel de tamaño 2 y como función de activación se utiliza Leaky ReLU [12]. El input de la red convolucional es la voxelización de la figura con resolución 32. Y el output de esta red es 3 planos de simetría y 3 ejes de rotación predichos. En principio este modelo no puede detectar más de 3 simetrías de cada tipo, más adelante se discutirá el cómo superar esta limitación.

2.2.1. Función de pérdida de simetrías de reflexión

Considerando la importancia que toma esta función de pérdida para el resto del informe se presenta la definición de ésta en el presente informe, la cual es extraída desde la definición original de PRS-Net y se enfoca a la predicción de simetrías de reflexión.

Sea $P_i = (n_i, d_i)$ los parámetros del i-ésimo plano en forma implícita, donde n_i es la dirección normal del plano de simetría $a_i x + b_i y + c_i z + d_i = 0$. La función de pérdida utilizada para entrenar las predicciones de planos de simetría de este modelo está dada por:

$$L = \frac{1}{H} \sum_{i=0}^{H-1} L_i^{SDE} + w_r L^{reg} \tag{2.3}$$

Donde L_i^{SDE} es el error por distancia de simetría (SDE por sus siglas en inglés) asociado al i-ésimo plano, w_r es el coeficiente de regularización, H la cantidad de planos de simetría predichos y L^{reg} es la pérdida de regularización. Esta última se define como:

$$L^{reg} = \left\| NN^T - I \right\|_F^2 \tag{2.4}$$

Donde $N = [\hat{n_1}, \hat{n_2}, \ldots, \hat{n_M}]$, $\hat{n_i}$ es la normal normalizada del i-ésimo plano y $\|\cdot\|_F$ es la norma de Frobenius. Notar que cuando N es ortogonal $L^{reg} = 0$ es decir que este término de regularización alcanza su mínimo cuando los planos de simetría predichos son ortogonales entre sí. Si bien esto puede parecer que va a forzar duramente que los planos sean ortogonales esto se puede regular variando w_r.

$$L_i^{SDE} = \sum_{j=0}^{|S|} \hat{D}_j^{(i)} \tag{2.5}$$

Donde S es la muestra de los puntos originales y $\hat{D}_j^{(i)}$ es el error de simetría *aproximado* para el punto S_j y el plano P_i. El error de simetría $D_j^{(i)}$ esta dado por:

$$D_j^{(i)} = \min_{s \in S} \left\| \text{reflected}(S_j^{(i)}, P_i) - s \right\| \tag{2.6}$$

Donde reflected$(S_j^{(i)}, P_i)$ es el j-ésimo punto de la muestra reflejado ante el i-ésimo plano predicho. Pero dado que esto es muy costoso de calcular, se hace una aproximación $\hat{D}_j^{(i)}$ que en vez de obtener la distancia con respecto a s se obtiene la menor distancia al centro del voxel más cercano.

2.3. SymmetryNet

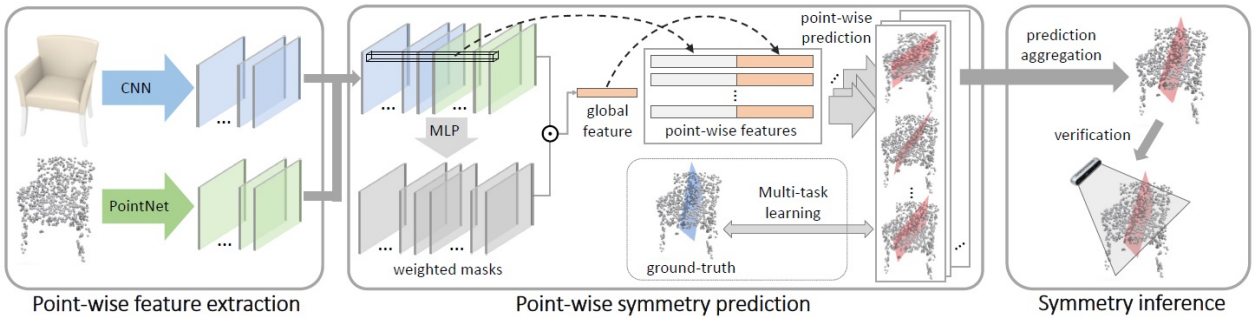

Figura 2.2: Resumen del método descrito por SymmetryNet. (Figura extraída desde la fuente original [6])

SymmetryNet [6] aprende a predecir simetrías de reflexión y rotación a partir de una imagen RGB-D de manera supervisada. Para hacer esto obtiene características de la imagen RGB usando una CNN y a partir del mapa de profundidad usando PointNet [9]. Con estas características crea un vector global que codifica las características encontradas en toda la imagen RGB-D y genera un vector de características locales para cada punto de la imagen. Luego, a partir de las características obtenidas para cada punto se sigue una estrategia de *Multi-task learning*. Las tareas a entrenar por cada punto son: clasificación del tipo de simetría, regresión de los parámetros de simetría, regresión de la posición del punto contraparte simétrico y clasificación sobre los puntos tal de encontrar su contraparte simétrica.

Este esquema de entrenar 4 tareas de manera simultánea para cada punto se realiza de esta manera para obligar a la red a aprender el concepto real de simetría y mejorar la generalización de la red. Sin embargo, hace mucho más complicado el proceso de entrenamiento y más costosa la inferencia.

Con respecto al proceso de entrenamiento de este modelo es importante notar que la función de pérdida se define según el tipo de simetría definida y para poder extender esta red a que prediga más de una simetría por figura se decide predecir un número M máximo de simetrías candidatas por punto. Para poder entrenar una red con M simetrías candidatas debe definirse una manera para relacionar las simetrías reales con las candidatas. Para relacionar las simetrías candidatas con las reales se resuelve un problema de asignación óptima [13] con una matriz de costos definida por la diferencia de aplicar la simetría candidata versus la real.

Como se puede ver este proceso es bastante complicado, para más detalle se recomienda revisar la fuente original [6].

2.4. SHREC 2023: Symmetry Detection on 3D Point Clouds

El dataset [7] corresponde a un track de *3D Shape Retrieval Challenge* con el objetivo de probar algoritmos de detección de simetrías de reflexión sobre nubes de puntos. Este está compuesto de 69,000 nubes de puntos con sus planos de simetría conocidos donde 60,000 pertenecen al conjunto de train y 9000 al conjunto de test. Estas nubes de puntos son generadas a partir de curvas las cuales tienen simetrías de reflexión conocidas y se transforman en nubes de puntos al proyectarlas a un cilindro o a un cono. Luego a estas nubes de puntos generadas se les aplica una perturbación lo que resulta con las nubes de puntos presentes en el dataset. Esto se puede revisar a más detalle en la Sección 2 de la fuente original [7]. Al momento de evaluar los métodos implementados se segmentará según tipo de curva y tipo de perturbación aplicada para poder estudiar como varía la eficacia del método ante estas variables.

Para poder entrenar los modelos se utiliza el conjunto de train de este dataset y se prueba sobre el conjunto de test usando las métricas de evaluación definidas en la sección 2.5. Para poder hacer esto se requiere que los modelos a probar reporten las simetrías detectadas con el mismo formato de las etiquetas, esto es, representar el plano según su normal y un punto. Además se requiere que se añada un valor de confianza que esté en el rango [0,1] el cual representa la confianza del modelo en que la simetría retornada es efectivamente una simetría del modelo.

2.5. Métricas de evaluación

Las métricas a utilizar son las siguientes:

- **PHC**: La precisión a la mayor confianza (PHC por sus siglas en inglés) corresponde al porcentaje de ejemplos donde la predicción de mayor confianza corresponde a un verdadero positivo. Esta métrica tiene por objetivo medir qué tan eficaz es un algoritmo para detectar simetrías con el mayor nivel de confianza. Para poder determinar cuando una predicción es verdadera positiva se utilizó el criterio de igualdad de la definición 1.

- **MAP**: Usando la definición de *mean average precision* [14] es una métrica utilizada para evaluar el rendimiento de un conjunto completo de simetrías. Al igual que con el PHC, se utiliza la definición de igualdad entre planos para determinar los verdaderos positivos.

Capítulo 3

Métodos implementados

Este capítulo describe los métodos implementados basados en PRS-Net y Symmetry-Net. Fue necesario aplicar modificaciones a ambos métodos debido a que el dataset utilizado presenta desafíos que en la postulación original de ellos no estaban presentes. Además, se decidió remover la predicción de simetrías de rotación de ambos modelos, ya que en el dataset utilizado no se tenía información de este tipo de simetrías. Se procuró efectuar modificaciones que mantuvieran la esencia de cada método por lo que se espera que los resultados obtenidos sean representativos de su efectividad. Las modificaciones realizadas a ambos métodos y los resultados obtenidos son detallados en el resto de este capítulo.

3.1. Modelo basado en PRS-Net

3.1.1. Limitaciones del método original frente al dataset de SHREC 2023

El modelo original de PRS-Net predice hasta 3 planos de simetría para una misma figura. Esto es un problema para utilizar este método con el nuevo dataset, ya que este cuenta con figuras que tienen mucho más de 3 simetrías. Para resolver esto se escogió aumentar la cantidad de cabezas de predicción a un número M que fuera mayor a la cantidad máxima de simetrías que una figura podía presentar. Sin embargo, para aplicar esta modificación se tuvo que ajustar también el coeficiente de regularización de la función de pérdida de PRS-Net para permitir a la red no castigar tanto predecir planos no ortogonales.

3.1.2. Descripción del método

El método está compuesto por 5 pasos presentados en la figura 3.1. El primer paso del método implementado es transformar la nube de puntos original a una representación volumétrica. Para hacer esto se escalan las nubes de puntos originales según la siguiente fórmula:

$$p_i = \frac{p_i - \min(p)}{\|\max(p) - \min(p)\|} \tag{3.1}$$

Donde p_i es el i-ésimo punto de la nube de puntos p. Con este escalamiento se deja a toda la nube de puntos en el intervalo $[0, 1]$. Luego se calcula la voxelización de resolución R donde el tamaño del voxel es $\frac{1}{R}$.

El siguiente paso consiste en la obtención de características a partir de la voxelización mediante el uso de una red convolucional. Esta se compone de 5 capas de convolución 3D con un kernel de tamaño 3, padding 1 y stride 1. Después de cada convolución se usa LeakyRelu [12] como función de activación y una capa de max pooling de kernel de tamaño 2 y stride 1.

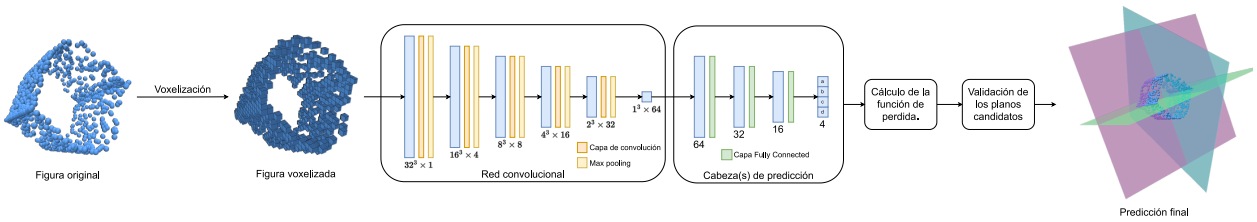

Figura 3.1: Resumen del método basado en PRS-Net.

La dimensión de input de esta red es la voxelización del modelo original con resolución $R = 32$ que tiene dimensiones 32^3 y el output es un vector de características de dimensión 64. Este vector actúa como el input de las cabezas de predicción. Una cabeza de predicción está compuesta de una red MLP de dimensión $(32, 16, 4)$ y de función de activación LeakyRelu que entrega como output los parámetros del plano de simetría. La cantidad de cabezas de predicción a utilizarse es un hiperparámetro del método y se denota como H.

Luego para cada cabeza H_i se debe calcular su error de distancia de simetría como función de pérdida denotada por L_i^{SDE}. Este se define en la ecuación 2.5 y requiere utilizar una muestra de los puntos del modelo original. Esta muestra de puntos se obtiene por muestreo aleatorio del modelo original en cada iteración del entrenamiento. Por otra parte se debe calcular la pérdida de regularización, esta se define en la ecuación 2.4 y asegura que la red no aprenda a predecir el mismo plano para todas las cabezas de predicción. La función de pérdida total está definida en la ecuación 2.3.

En los experimentos en un momento se decide reemplazar la función que se utiliza para calcular el error de distancia de simetría L_i^{SDE} por la función de pérdida de Chamfer L_i^{CD}. Esta se define de la siguiente manera:

$$L_i^{CD} = D_{CD}(S, \text{reflected}(S, P_i)) \tag{3.2}$$

Donde S es una muestra de los puntos originales, D_{CD} es la distancia de Chamfer definida en la ecuación 2.2 y reflected(S, P_i) es la nube de puntos S reflejada sobre el plano P_i que es el predicho por la cabeza H_i. Este cambio se realiza de esta manera, ya que lo que se intenta aproximar con el error de distancia de simetría es el cambio de la

nube de puntos una vez aplicada la transformación de la simetría. Y dado que la distancia de Chamfer entrega una mejor aproximación de esto resulta una atractiva opción a probar.

Finalmente, se utiliza un proceso simple de validación para las H predicciones resultantes. Esta tiene por objetivo remover planos que claramente no representan una simetría y remover planos de predicción repetidos dejando el que tiene mayor evidencia de ser el verdadero plano de simetría. Para esto se utiliza el error de distancia de simetría, primero se define una tolerancia γ tal que para toda predicción H_i que cumpla que $L_i^{SDE} > \gamma$ es inmediatamente descartada. Luego se remueven los duplicados buscando para cada par de planos si es que el ángulo entre ellos es menor a ϕ se remueve el de mayor L_i^{SDE}. Después de este proceso se ordenan por L_i^{SDE} y se define la confianza de la predicción como el resultado de normalizar usando el método de MinMax [15] los errores de distancia de simetría.

3.1.3. Resultados obtenidos

Se llevan los siguientes experimentos:

- **(E1)** PRS-Net con hiperparámetros originales: Se prueba en primera instancia un modelo con los hiperparámetros originales extraídos de la fuente original. Esto es, utilizar la cantidad de cabezas $H = 3$, el coeficiente de regularización $w_r = 25$, la resolución de la voxelización $R = 32$, batch size 32, tamaño de la muestra de los puntos originales $|S| = 1000$ y la función para calcular el error de distancia de simetría original. Para el proceso de validación se utilizan hiperparámetros adaptados al dataset actual, esto es, usar $\gamma = 0.5$ y $\phi = 10$º. Este experimento actúa como *baseline* para el resto. Notar que dado que el dataset tiene figuras con más de 3 simetrías se espera que este modelo tenga malos resultados.

- **(E2)** PRS-Net con mayor H: Se decide aumentar la cantidad de cabezas de predicción a $H = 27$ y disminuir el coeficiente de regularización $w_r = 5$. Se debe disminuir w_r para evitar que el modelo converja a retornar los planos más ortogonales posibles. La cantidad de cabezas escogida es mayor que la cantidad máxima de simetrías que presenta una figura en el dataset. El resto de los hiperparámetros se mantiene de (E1).

- **(E3)** PRS-Net utilizando la función de pérdida de Chamfer L_i^{CD}: En este experimento se busca probar si reemplazar la función de pérdida por error de distancia de simetría por la función de pérdida de Chamfer vale la pena. Se utiliza la función definida en la ecuación 3.2, se aumenta aún más el número de cabezas a $H = 32$ y se disminuye el coeficiente de regularización aún más a $w_r = 1.0$. Además se modifica el tamaño de las muestras a $|S| = 2048$. Para el proceso de validación se modifican los parámetros para la nueva función de pérdida, esto es, usar $\gamma = 0.001$ y $\phi = 10$º.

- **(E4)** Prueba similar a (E3) con otros hiperparámetros: En este experimento se toma como base el tercer experimento y se entrena durante más tiempo comenzando

con un coeficiente de regularización $w_r = 5.0$ el cual se disminuye a 1.0 pasadas 27 épocas y luego se vuelve a disminuir a 0.001 después de la época 38. Se espera que al utilizar un coeficiente de regularización alto en un comienzo el modelo evite tener predicciones duplicadas y luego al bajar la importancia de la regularización el modelo se enfoque en aprender simetrías reales mejorando los resultados. Se presentan los resultados por cada etapa del entrenamiento de este experimento (E4.1, E4.2 y E4.3).

- **(E5)** H pequeño y w_r pequeño: Se utiliza $H = 6$, $w_r = 0.5$, batch size 64, se utiliza la función de pérdida original y |S| = 1024. Este experimento busca probar si se obtenían mejores resultados al plantear un modelo más simple.

Los resultados obtenidos se presentan en la tabla 3.1. Estos resultados prueban que este modelo no es capaz de recuperar los planos de simetría en figuras que tienen más de 3 simetrías. Para intentar entender mejor el rendimiento del modelo se evalúan las métricas con 3 combinaciones distintas de ϵ y θ para bajar la exigencia para considerar dos planos iguales según la definición 1. El intervalo estricto indica que se utiliza como $\epsilon = 1\,\%$ de la diagonal de la nube de puntos y $\theta = 1$º. Los otros intervalos se definen de la misma manera.

Tabla 3.1: Resultados experimentos PRS-Net.

	Estricto (1 % ; 1 º)		**Relajado (3 % ; 3º)**		**Extremo (5 % ; 5º)**	
Experimento	**MAP**	**PHC**	**MAP**	**PHC**	**MAP**	**PHC**
E1	0,02	0,01	0,07	0,05	0,16	0,12
E2	0,04	0,01	0,13	0,07	0,25	0,15
E3	**0,05**	**0,03**	0,15	0,09	0,27	0,18
E4.1	0,04	**0,03**	**0,16**	0,11	0,29	0,21
E4.2	0,04	**0,03**	**0,16**	**0,13**	**0,31**	**0,25**
E4.3	0,01	0,01	0,14	0,10	0,28	0,20
E.5	0,02	0,01	0,11	0,08	0,24	0,18

El dataset de SHREC 2023 permite evaluar las métricas según el tipo de figura y según perturbación aplicada. Esto permite evaluar a más detalle la eficacia del modelo y su robustez ante cada tipo de curva o perturbación. Las definiciones detalladas de los tipos de curva y las perturbaciones aplicadas se encuentran en la fuente original. A modo de análisis se escoge estudiar los resultados detallados del modelo resultante del experimento 3 dado que este fue el que obtuvo las mejores métricas en el intervalo estricto. En la siguiente tabla se presentan las métricas obtenidas según el tipo de perturbación:

Tabla 3.2: Detalle de las métricas obtenidas por el modelo resultado del experimento 3 según el tipo de perturbación.

Perturbación Aplicada	**MAP**	**PHC**
Sin perturbación	0,05	0,03
Ruido uniforme	0,05	0,03
Ruido Gaussiano	0,04	0,02
Submuestreado	0,05	0,03
Ruido uniforme y submuestreado	0,04	0,02
Ruido Gaussiano y submuestreado	0,05	0,03

Se observa que el método desarrollado no tiene diferencias de eficacia relevantes según el tipo de perturbación aplicada. Esto es esperable porque la voxelización resultante de una figura con ruido es muy similar a la de una figura sin ruido. Luego, los resultados obtenidos por tipo de figura son:

Tabla 3.3: Detalle de las métricas obtenidas por el modelo resultado del experimento 3 según el tipo de figura.

Tipo de figura	**MAP**	**PHC**
M convexities	0,04	0,02
Mouth curve	**0,09**	**0,04**
Egg Keplero	0,01	0,01
Lemniscate Bernoulli	0,06	0,03
Citrus	0,03	0,02
Astroid	**0,09**	**0,05**
Geometric petal	0,02	0,01

Al contrario de lo encontrado con el análisis según tipo de perturbación, las métricas obtenidas por tipo de figura si presentan diferencias interesantes. En la tabla 3.3 se destaca que para las clases Mouth curve y Astroid se obtienen resultados bastantes mejores que para el resto de las otras clases. Se cree que esto es debido a que la representación volumétrica obtenida de las figuras de estos tipos presentan patrones que permiten detectar su simetría de mejor manera.

Finalmente, a pesar de todos los experimentos, no se logra encontrar una configuración que permita que este método obtenga buenos resultados. Se piensa que esto se debe a que se pierde demasiada información en transformar las nubes de puntos a representaciones volumétricas. Sin embargo, se encuentra mucho valor en que este método es de aprendizaje no supervisado por lo que se puede intentar utilizar sobre figuras que cumplan de mejor manera la hipótesis que plantea el método original que es que sólo existan a lo más 3 simetrías por figura.

3.2. Modelo basado en SymmetryNet

3.2.1. Resumen de las modificaciones realizadas

Para poder evaluar la eficacia de SymmetryNet en el dataset de SHREC 2023 es necesario modificar varios aspectos del método original. El aspecto más relevante es cambiar el tipo de información que recibe SymmetryNet. En el modelo original se utiliza como input una imagen RGB-D mientras que con el dataset nuevo se debe utilizar una nube de puntos. Esto hace necesario modificar la arquitectura de SymmetryNet, en donde se utilizaba una capa de Spatially Weighted Pooling [16] para poder correlacionar la imagen de cada pixel de la imagen RGB con su contraparte en el mapa de profundidad. Esto se hacía para poder generar un vector de características por pixel y un vector de características globales para cada input.

Ya que el problema propuesto por SHREC 2023 considera nubes de puntos es necesario poder aplicar algún método de extracción de características que esté diseñado para manejar las particularidades de este tipo de dato. Es por eso que se decide reemplazar este paso inicial con un PointNet que obtenga las características de las nubes de puntos. Además este es el método que se ocupa para generar las características del mapa de profundidad por lo que se puede confirmar que este es el paso correcto.

El segundo aspecto que se modifica es el esquema de predicción del modelo original. Este se encuentra explicado a detalle en la sección 2.3. Se cambia el esquema de predicción denso a predecir un plano por cabeza de predicción. Esto se realiza para simplificar el proceso de entrenamiento, ya que el costo de entrenar por punto según la función de pérdida de la fuente original resulta prohibitivamente caro y en las pruebas preliminares la red no logra converger.

3.2.2. Descripción del método

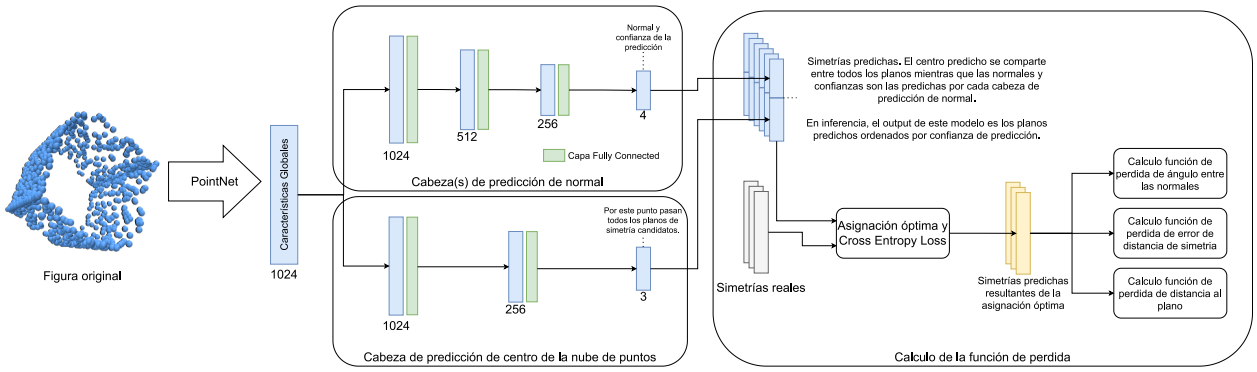

Figura 3.2: Resumen del método basado en SymmetryNet.

Este método obtiene características globales de las nubes de puntos utilizando PointNet las cuales son utilizadas para predecir el centro de la nube de puntos y una cantidad M de normales y confianzas. El predecir un sólo centro para todas las simetrías se reali-

za de esta manera por 2 motivos. El primero es que el separar esto hace más estable la convergencia de la red. El segundo, dado cómo se construyo el dataset de SHREC 2023 este cumple que todas las simetrías de una figura tienen el mismo centro, así que es una simplificación razonable. Después de haber obtenido los planos predichos se calcula la función de pérdida que está dada por la siguiente ecuación:

$$L = L^{confidence} + L^{angle} + L^{distance} + w_{SDE} L^{SDE} \tag{3.3}$$

Donde $L^{confidence}$ es la pérdida asociada a la confianza sobre las predicciones, L^{angle} es la pérdida asociada a los ángulos, $L^{distance}$ se asocia a las distancias, w_{SDE} es el hiperparámetro que modera que tanta importancia se le da a L^{SDE} que es la pérdida asociada al error de distancia de simetría. Para $L^{confidence}$ se debe poder encontrar una manera de relacionar las M predicciones con las K simetrías reales que puede tener una figura. Para hacer esto se aplica la solución propuesta en el modelo original que consiste en resolver un problema de asignación óptima entre ambos conjuntos el cual se define a continuación.

Sea Π una matriz de permutación con $\Pi_{m,k} \in \{0, 1\}$ indicando si la simetría k-ésima de verdad coincide con la simetría m-ésima predicha. Sea B una matriz de beneficio en la que $B_{m,k}$ la que representa el beneficio de emparejar la simetría k-ésima de verdad con la simetría m-ésima predicha. Una mayor similitud entre dos simetrías resulta en un beneficio más grande. Queda entonces el problema de optimización a resolver:

$$\arg\max_{\Pi} \prod_{m=1}^{M} \prod_{k=1}^{K} B_{m,k}^{\Pi_{m,k}} \tag{3.4}$$

$$\text{s.a.} \quad \sum_{m=1}^{M} \Pi_{m,k} = 1, \quad k \in \{1, \ldots, K\}; \quad \sum_{k=1}^{K} \Pi_{m,k} \leq 1, \quad m \in \{1, \ldots, M\}. \tag{3.5}$$

Para definir B se busca que el ángulo predicho y el real sea casi 0, ya que la predicción del centro de la nube de puntos asegura que la distancia con el plano sea pequeña. Por lo que se tiene que B se define como:

$$B_{m,k} = \left| \hat{n_m} \cdot \tilde{n_k}^T \right| - 1 \tag{3.6}$$

Una vez se ha encontrado la asignación óptima de planos predichos a planos reales se crea un vector One-hot O de largo M donde $O_i = 1$ si el i-ésimo plano predicho fue asignado a un plano real. Además, sea $\hat{C}$ el vector de largo M de confianzas predichas por el modelo. Luego se tiene que:

$$L^{confidence} = \text{LogLoss}(\hat{C}, O) \tag{3.7}$$

En resumen, $L^{confidence}$ es la entropía cruzada entre el vector de confianzas predicho

y el vector One-Hot que indica cuáles planos se asignan a un plano real. El resto de los términos de la función de pérdida se calculan sólo para aquellos planos predichos que fueron asignados a un plano real, denotemos estos planos como $\hat{y}_i = (\hat{n}_i, \hat{c})$ con $i \in [0, K]$ donde $\hat{n}_i$ sea la normal de ese plano y c sea el centro predicho que se comparte entre todos los planos. Sea el plano real $y_i = (n_i, c_i)$ aquel que se asignó a $\hat{y}_i$ donde n_i es normal y c_i el punto en ese plano. Entonces L^{angle} se define como:

$$L^{angle} = \frac{1}{K} \sum_{i=0}^{K} 1 - |\hat{n}_i \cdot n_i| \tag{3.8}$$

Continuando con la misma notación para referirnos al plano predicho y su plano real asignado se tiene que $L^{distance}$ se define como:

$$L^{distance} = \frac{1}{K} \sum_{i=0}^{K} \|\hat{c} - c_i\| \tag{3.9}$$

Y finalmente, L^{SDE} está dado por:

$$L^{SDE} = \frac{1}{K} \sum_{i=0}^{K} SDE(P, \hat{y}_i, y_i) \tag{3.10}$$

Donde P son los puntos del input y la función SDE calcula el error de distancia de simetría y se define como:

$$SDE(P, \hat{y}_i, y_i) = \frac{1}{|P|} \sum_{j=0}^{|P|} \|\text{reflected}(p_i, y_i) - \text{reflected}(p_i, \hat{y}_i)\| \tag{3.11}$$

En donde la función *reflected(p_i, y_i)* indica aplicar la simetría de reflexión al punto p_i bajo el plano y_i. Con esto se concluye la definición de la función de pérdida del método basado en SymmetryNet. Finalmente, la predicción del modelo en inferencia es simplemente las M simetrías predichas ordenadas por su confianza.

3.2.3. Resultados obtenidos

Para probar el método desarrollado para SymmetryNet se entrena el modelo usando $M = 27$ y $w_{SDE} = 0.1$. La cantidad de cabezas de predicción es mayor a la cantidad máxima de simetrías que una figura del dataset puede tener. Se entrena a lo largo de 4 épocas con batch size 1. Se escoge utilizar este tamaño de batch size, ya que en las pruebas demostró obtener los mejores resultados al no tener que tomar ninguna muestra de la nube de puntos de la figura. Este hecho es interesante pues se encontró que el método de muestreo de la nube de puntos original es muy importante para el rendimiento de este método.

Para escoger el valor del coeficiente de regularización $w_{SDE} = 0.1$ se buscó que la magnitud de la perdida asociada al SDE estuviera dentro del mismo orden de magnitud

que el resto de los miembros de la función de pérdida. Esto se realizó de esta manera para asegurar que el modelo no sólo aprendiera a regresar la normal y el centro por separado, sino que buscara aprender el concepto real de simetría por medio de la pérdida por error de distancia de simetría.

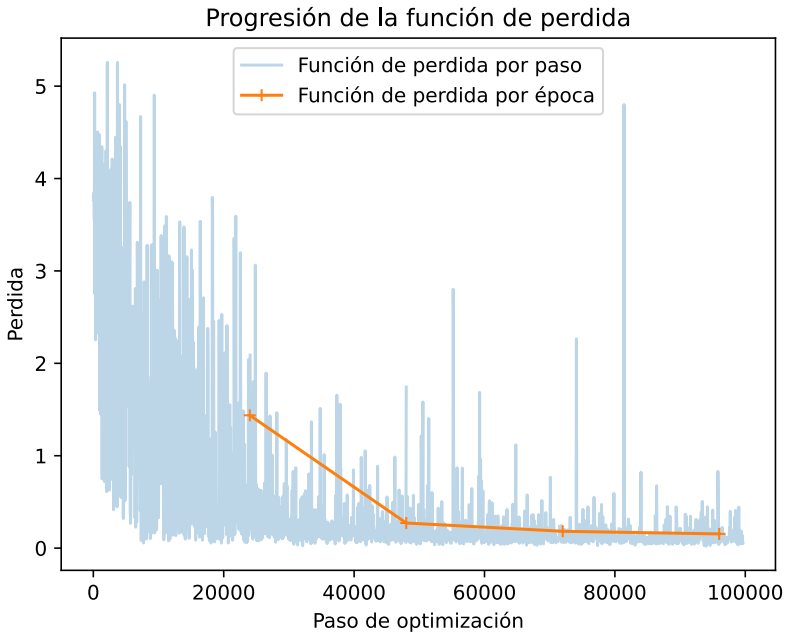

Figura 3.3: Progresión de la función de pérdida a través del entrenamiento.

En la figura 3.3 se presenta la progresión de la función de pérdida a través del entrenamiento. Se aprecia el efecto de utilizar batch size 1, ya que la función de pérdida en cada paso varía en gran medida al comienzo del entrenamiento, pero luego al alcanzar la convergencia esta varianza disminuye de gran manera. El valor de la función de pérdida al final de cada época es el promedio de la pérdida para cada figura del dataset de entrenamiento. Por otro lado, el valor de la función de pérdida por paso es calculado en base sólo al ejemplo del paso de optimización en cuestión. También se observa que al final del entrenamiento se ha alcanzado la convergencia.

En la siguiente tabla se presentan las métricas obtenidas por este método sobre todo el dataset ocupando el criterio de igualdad entre planos presentado en la definición 1 con $\theta = 1^{\text{o}}$ y $\epsilon = 1\,\%$ de la diagonal de la nube de puntos.

Tabla 3.4: Resultados obtenidos por el método basado en SymmetryNet.

MAP	**PHC**
0,773	0,695

Al igual que con el análisis de resultados de PRS-Net, en las tablas 3.5 y 3.6 se presentan los resultados segmentados según el tipo de perturbación aplicada y el tipo de

figura respectivamente. Se continúa utilizando los mismos parámetros para determinar la igualdad entre planos.

Tabla 3.5: Resultados obtenidos por el método basado en SymmetryNet según tipo de perturbación aplicada.

Perturbación Aplicada	**MAP**	**PHC**
Sin perturbación	0,771	0,691
Ruido uniforme	0,771	0,688
Ruido Gaussiano	0,781	0,703
Submuestreado	0,779	0,698
Ruido uniforme y submuestreado	0,764	0,689
Ruido Gaussiano y submuestreado	0,778	0,706

La tabla 3.5 permite apreciar que el método implementado es robusto ante perturbaciones del input. Los resultados no varían de manera significativa ni para MAP o PHC. Sin embargo, en la siguiente tabla se puede observar claramente que el método no tiene la misma eficacia para todos tipos de figura. Para los dos primeros tipos de figura, M convexities y Geometric petal, los resultados obtenidos son mucho peores que para el resto.

Tabla 3.6: Resultados obtenidos por el método basado en SymmetryNet según tipo de figura.

Tipo de figura	**MAP**	**PHC**
M convexities	0,422	0,329
Geometric petal	0,509	0,363
Astroid	0,741	0,431
Egg Keplero	0,805	0,756
Mouth curve	0,966	0,999
Citrus	0,969	0,991
Lemniscate Bernoulli	0,993	0,998

La dificultad de este método para poder predecir los planos de simetría en esos tipos de figuras se atribuye a la predicción de los vectores normales. Al evaluar la tarea de predicción del centro de la nube de puntos por separado se obtiene $MAP = 1.0$ lo que indica que el modelo es más que capaz de regresar de manera perfecta el centro de la nube de puntos. Para estudiar a más detalle la calidad de las predicciones de las normales se utilizan distintos valores de θ para aumentar la tolerancia a la predicción de planos de simetría imprecisos y se mantiene el valor de $\epsilon = 1\,\%$ de la diagonal de la nube de puntos. Estos resultados se pueden observar en la figura 3.4.

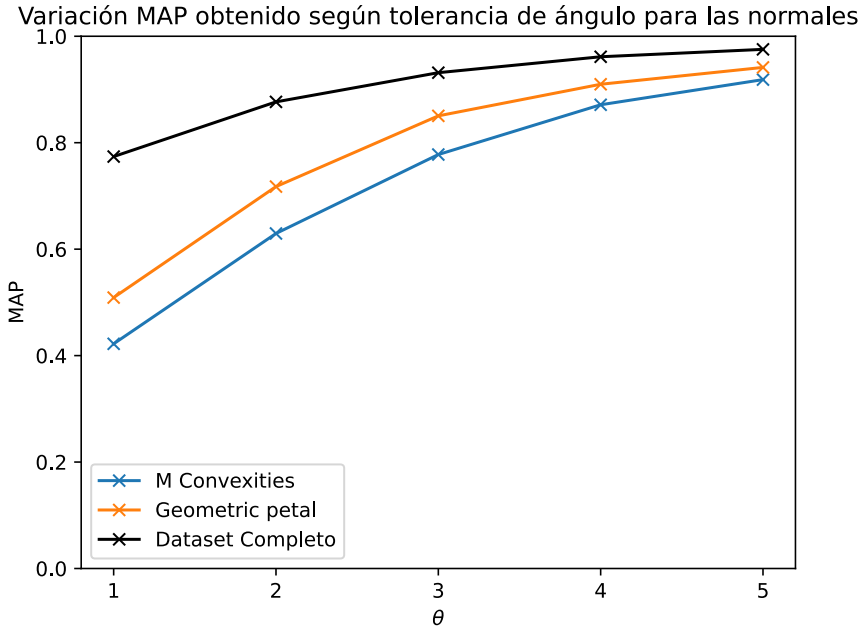

Figura 3.4: MAP obtenido según distintos valores de θ.

Se aprecia que el método es capaz de detectar las simetrías existentes en las figuras de los tipos problemáticos, pero falla al determinarlas con precisión. En la figura 3.5 se presenta una comparación los planos predichos y los reales para una figura de tipo *M Convexities*.

(a) Planos reales

(b) Planos predichos

Figura 3.5: Ejemplo de predicción imprecisa. Se detecta una cantidad correcta de planos y las normales detectadas son cercanas a las reales. Sin embargo, al momento de evaluar las métricas sólo 2 de 5 planos efectivamente son detectados como verdaderos positivos. Para ayudar a la visualización se descartaron las simetrías predichas cuya confianza es muy cercana a 0.

Capítulo 4

Conclusiones

El principal aporte de este informe es la implementación y evaluación de dos métodos de detección de simetrías de reflexión en nubes de puntos. Al haber sido ambos métodos adaptados para un dataset en común se logra hacer una comparación efectiva entre ambos modelos para poder concluir cuál es el que obtiene mejores resultados.

Con respecto al primer método, se utiliza como base lo descrito en PRS-Net para poder obtener los planos de simetrías a partir de nubes de puntos de manera no supervisada. Sin embargo, esta técnica no obtiene buenos resultados en el dataset utilizado. Se cree que esto es debido a que el método original de PRS-Net está diseñado para detectar simetrías de reflexión en objetos que presentan a lo más 3 simetrías, mientras que en el dataset utilizado las figuras pueden presentar muchas más simetrías. Se prueba con una función de pérdida modificada que es, en teoría, más precisa, pero esto no se traduce en mejores resultados por lo que se concluye que este método es incapaz de obtener mejores resultados para este dataset en particular.

Otra teoría que se tiene para explicar los resultados obtenidos por el primer método es que el hecho de transformar la nube de puntos a una representación volumétrica conlleva una gran pérdida de información. Esta pérdida de información resulta en que la red no pueda aprender los patrones necesarios para obtener mejores resultados. Es por este motivo que también se cree que el segundo método obtuvo mejores resultados.

Por otra parte, el segundo método se inspira en lo descrito por SymmetryNet. En este método se utiliza una red PointNet para extraer características desde una nube de puntos y a partir de ellas se predicen los planos de simetrías. Para poder predecir una cantidad arbitraria de planos de simetría por figura se hace uso de una estrategia basada en la asignación óptima para aprender a que el modelo indique cuál es la confianza asociada a cada predicción. Una particularidad encontrada es la mejora de rendimiento presentada por este método al momento de entrenar con batch size 1 sin tomar muestras de las nubes de puntos. Se cree que esto se debe a que el utilizar muestreo aleatorio sobre nubes de puntos no es apropiado en este tipo de problemas.

Se ha encontrado que los métodos desarrollados son robustos a perturbaciones en el input y que existen figuras las cuales son más difíciles de encontrar sus planos de sime-

trías. Se encuentra que estas figuras presentan simetrías las cuales pueden ser detectadas por el segundo método mas éste no es capaz de recuperarlas de manera precisa. Estos resultados hacen sentido pues para ambos métodos se espera que la representación aprendida sea robusta a las perturbaciones, pero es esperable que existan tipos de figuras que son más difíciles de presentar por ambos métodos.

Por último, con los resultados obtenidos en este informe se ha logrado cumplir los objetivos presentados en la sección 1.1. En particular, se ha logrado vislumbrar que el método basado en SymmetryNet obtiene mejores resultados para esta tarea. Y además se han postulado posibles motivos para explicar los resultados obtenidos.

4.1. Trabajo futuro

Durante el desarrollo y evaluación de ambos métodos se han encontrado fenómenos que motivan explorar como ciertas variables afectan el rendimiento de cada método. Una variable que ha probado su impacto en los resultados obtenidos por cada método es la estrategia de muestreo utilizado, se ha encontrado que utilizar muestreo aleatorio no parece ser lo más óptimo. Sería interesante explorar qué rendimiento se obtendría por cada modelo al aplicar otras estrategias de muestreo diseñadas para ser usadas en nube de puntos.

Otra variable interesante a explorar es la resolución de la voxelización calculada en el primer método. Es posible que al utilizar una mayor resolución se pudieran obtener mejores resultados. Sin embargo, es importante balancear la resolución con el costo computacional asociado al utilizarla. Ya al ocupar una resolución de 32^3 se ha tenido que calcular previamente las representaciones volumétricas de todo el dataset lo que ha tenido un costo no menor.

Como se mencionó previamente, el segundo método utiliza una red para poder extraer características globales de los puntos. Sería interesante probar otras arquitecturas para evaluar si se pueden mejorar aún más los resultados obtenidos en esta instancia. Se cree fuertemente que este es el caso y el método está diseñado para que aplicar estas modificaciones no conlleve mayor dificultad.

Finalmente, resulta interesante además probar como generalizan estos métodos para tipos de figuras no presentes en el conjunto de entrenamiento. Esto con el objetivo de comprobar si el método está aprendiendo efectivamente el concepto de simetría.

Bibliografía

[1] S. Wu, C. Rupprecht, and A. Vedaldi, "Unsupervised learning of probably symmetric deformable 3d objects from images in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1–10, 2020.

[2] J. Jiang, Z. Chen, and K. He, "A feature-based method of rapidly detecting global exact symmetries in cad models," *Computer-Aided Design*, vol. 45, p. 1081–1094, 08 2013.

[3] Y. Brhane Hagos, A. Gubern Mérida, and J. Teuwen, "Improving breast cancer detection using symmetry information with deep learning," in *Image Analysis for Moving Organ, Breast, and Thoracic Images* (D. Stoyanov, Z. Taylor, B. Kainz, G. Maicas, R. R. Beichel, A. Martel, L. Maier-Hein, K. Bhatia, T. Vercauteren, O. Oktay, G. Carneiro, A. P. Bradley, J. Nascimento, H. Min, M. S. Brown, C. Jacobs, B. Lassen-Schmidt, K. Mori, J. Petersen, R. San José Estépar, A. Schmidt-Richberg, and C. Veiga, eds.), (Cham), pp. 90–97, Springer International Publishing, 2018.

[4] W. Qiu, J. Yuan, E. Ukwatta, Y. Sun, M. Rajchl, and A. Fenster, "Prostate segmentation: an efficient convex optimization approach with axial symmetry using 3-d trus and mr images," *IEEE transactions on medical imaging*, vol. 33, no. 4, pp. 947–960, 2014.

[5] L. Gao, L.-X. Zhang, H.-Y. Meng, Y.-H. Ren, Y.-K. Lai, and L. Kobbelt, "Prs-net: Planar reflective symmetry detection net for 3d models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 6, pp. 3007–3018, 2020.

[6] Y. Shi, J. Huang, H. Zhang, X. Xu, S. Rusinkiewicz, and K. Xu, "Symmetrynet: learning to predict reflectional and rotational symmetries of 3d shapes from single-view rgb-d images," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–14, 2020.

[7] I. Sipiran, C. Romanengo, B. Falcidieno, S. Biasotti, G. Arvanitis, C. Chen, V. Fotis, J. He, X. Lv, K. Moustakas, S. Peng, I. Romanelis, W. Sun, C. Vlachos, Z. Wu, and Q. Xie, "SHREC 2023: Detection of Symmetries on 3D Point Clouds Representing Simple Shapes," in *Eurographics Workshop on 3D Object Retrieval* (U. Fugacci, G. Lavoué, and R. C. Veltkamp, eds.), The Eurographics Association, 2023.

[8] N. J. Mitra, M. Pauly, M. Wand, and D. Ceylan, "Symmetry in 3d geometry: Extraction and applications," *Computer Graphics Forum*, vol. 32, no. 6, pp. 1–23, 2013.

[9] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017.

[10] D. Maturana and S. Scherer, “Voxnet: A 3d convolutional neural network for real-time object recognition,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928, 2015.

[11] H. Fan, H. Su, and L. Guibas, “A point set generation network for 3d object reconstruction from a single image,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 2463–2471, IEEE Computer Society, jul 2017.

[12] A. L. Maas, “Rectifier nonlinearities improve neural network acoustic models,” 2013.

[13] D. F. Crouse, “On implementing 2d rectangular assignment algorithms,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1679–1696, 2016.

[14] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *International Journal of Computer Vision*, vol. 111, pp. 98 – 136, 2014.

[15] S.Gopal, K. Patro, and K. K. Sahu, “Normalization: A preprocessing stage,” *ArXiv*, vol. abs/1503.06462, 2015.

[16] Q. Hu, H. Wang, T. Li, and C. Shen, “Deep cnns with spatially weighted pooling for fine-grained car recognition,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3147–3156, 2017.