Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas

Aprendizaje de principio a fin de estimación de pose 6D de objetos utilizando correspondencias densas y *Perspective-n-Point* diferenciable

T E S I S

que para optar por el grado de

Maestro en Ciencia e Ingeniería de la Computación

PRESENTA:
Dennis Alberto Mendoza Solís

Tutor Principal:
Dr. Gibrán Fuentes Pineda,
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas

Ciudad Universitaria, Cd. Mx., Mayo 2022

Agradecimientos

A mis padres y a mis hermanas, por apoyarme y amarme siempre en cada paso a pesar de las adversidades. No habría podido llegar hasta aquí sin ustedes. Gracias por todo.
A Vivian Bass, mi compañera de vida y apoyo incondicional. Gracias por estar a mi lado todo este tiempo, por compartir tantas experiencias juntos y por motivarme siempre.
A mi tutor y amigo, el Dr. Gibrán Fuentes, por todo el conocimiento y apoyo que me brindó durante la maestría para poder realizar esta tesis.
A todos los miembros del Grupo Golem con los que tuve la oportunidad de colaborar, en especial al Dr. Luis Pineda y al Dr. Caleb Rascón, por haber creído en mí y por haberme mostrado el mundo de la robótica y la inteligencia artificial, así como sus capacidades y limitaciones. Gracias por haberme dado la oportunidad de trabajar en algo que me apasiona.
Al Consejo Nacional de Ciencia y Tecnología y a los mexicanos, por haberme otorgado una beca sin la cual quizá no habría podido estudiar la maestría.
A la Universidad Nacional Autónoma de México, por haberme dado la oportunidad de pertenecer a su comunidad durante mi estancia en la Escuela Nacional Preparatoria 6, la Facultad de Ingeniería y el Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas. En muy pocos países se tiene la oportunidad de recibir una educación de tan buena calidad prácticamente sin costo.

Resumen

En esta tesis se presenta un nuevo método para la estimación de pose 6D de objetos utilizando correspondencias densas y *Perspective-n-Point* diferenciable. A diferencia de métodos como DPOD (*Dense Pose Object Detector*) [24], el método propuesto realiza el aprendizaje de principio a fin. La arquitectura propuesta tiene un codificador-decodificador que posteriormente se conecta a una capa DSNT (*Differentiable Spatial to Numerical Transform*) [15], la cual obtiene correspondencias entre el modelo del objeto y la imagen de entrada. Finalmente, dadas esas correspondencias, ocupamos un módulo BPnP (*Backpropagatable Perspective-n-Point*) [4] que calcula la pose del objeto. Los resultados muestran un rendimiento comparable a arquitecturas como YOLO6D [22], PVNet (*Pixel-wise Voting Network*) [16] y DPOD sobre el conjunto de datos LINEMOD [10] utilizando la métrica ADD (*Average Distance*).

Abstract

In this thesis, a new method for estimating the 6D pose of objects using dense correspondences and differentiable *Perspective-n-Point* is presented. Unlike methods such as DPOD (*Dense Pose Object Detector*) [24], the proposed method performs end-to-end object pose estimation. The net architecture has an encoder-decoder that is later connected to a DSNT (*Differentiable Spatial to Numerical Transform*) [15] layer, which is responsible for obtaining correspondences between the object model and the input image. Finally, given these correspondences, we use a BPnP (*Backpropagatable Perspective-n-Point*) [4] module that computes the pose of the object. The results show comparable performance to architectures such as YOLO6D [22], PVNet (*Pixel-wise Voting Network*) [16] and DPOD on the LINEMOD dataset [10] using the ADD (*Average Distance*) metric.

Índice general

Notación

Números y arreglos

a	Un escalar (entero o real)
$\boldsymbol{a}$	Un vector
$\boldsymbol{A}$	Una matriz
A	Un tensor
$\boldsymbol{I}_n$	Matriz identidad con n renglones y n columnas

Conjuntos y gráficas

$\mathbb{A}$	Un conjunto
$\mathbb{R}$	El conjunto de los números reales
$[a, b]$	El intervalo real incluyendo a y b
$(a, b]$	El intervalo real excluyendo a pero incluyendo b
$\mathcal{G}$	Una gráfica

Indexado

a_i	Elemento i del vector $\boldsymbol{a}$, con indexado comenzando en 1
$A_{i,j}$	Elemento i, j de la matriz $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Renglón i de la matriz $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Columna i de la matriz $\boldsymbol{A}$
$\mathsf{A}_{i,j,k}$	Elemento (i, j, k) del tensor de 3 dimensiones A

Operaciones de álgebra lineal

$\boldsymbol{A}^\top$	Transpuesta de una matriz $\boldsymbol{A}$
$\det(\boldsymbol{A})$	Determinante de $\boldsymbol{A}$

Cálculo

$\frac{dy}{dx}$	Derivada de y con respecto a x
$\frac{\partial y}{\partial x}$	Derivada partical de y con respecto a x
$\nabla_{\boldsymbol{x}} y$	Gradiente de y con respecto a $\boldsymbol{x}$

Funciones

$f : \mathbb{A} \to \mathbb{B}$	La función f con dominio $\mathbb{A}$ y rango $\mathbb{B}$
$f \circ g$	Composición de las funciones f y g
$\log x$	Logaritmo natural de x
$\|\|\boldsymbol{x}\|\|_p$	Norma L^p de $\boldsymbol{x}$
$\|\|\boldsymbol{x}\|\|$	Norma L^2 de $\boldsymbol{x}$

Conjuntos de datos y distribuciones

p_{data}	La distribución de generación de datos
$\hat{p}_{\text{data}}$	La distribución empírica definida por el conjunto de entrenamiento
$\mathbb{X}$	Un conjunto de muestras de entrenamiento
$\boldsymbol{x}^{(i)}$	La i-ésima muestra de un conjunto de datos de entrenamiento
$y^{(i)}$ or $\boldsymbol{y}^{(i)}$	El objetivo asociado con $\boldsymbol{x}^{(i)}$ para aprendizaje supervisado
$\boldsymbol{X}$	La matriz de dimensiones $m \times n$ con el ejemplo $\boldsymbol{x}^{(i)}$ en la fila $\boldsymbol{X}_{i,:}$

Capítulo 1

Introducción

La estimación de pose consiste en que dada una imagen donde hay un objeto de interés, se determine la posición relativa entre dicho objeto y la cámara desde la cual fue tomada la imagen. Suponiendo que se conocen los parámetros intrínsecos de la cámara, la pose describe la rotación y traslación del sistema de referencia de la cámara respecto a un sistema de referencia centrado en el objeto de interés, como se observa en la Figura 1.1. Esto se representa a través de una matriz de transformación geométrica tridimensional con seis grados de libertad (de ahí el nombre de estimación de pose 6D), tres de ellos correspondientes a la traslación y otros tres a la rotación [16].

La pose de un objeto se vuelve relevante en tareas como la manipulación robótica, donde el robot debe posicionar un manipulador en las coordenadas del objeto de interés para poder tomarlo. Conocer la pose es especialmente importante cuando las características geométricas del objeto requieren que sea manipulado de una parte específica, como la manija de una puerta o el asa de una bolsa. En estas circunstancias, estimar la pose de los objetos permite al robot manipularlos correctamente y mejorar la forma en que interactúa con su entorno.

La estimación de pose no está limitada a transformaciones geométricas en el mundo real, también puede ser utilizada para localizar adecuadamente objetos en ambientes virtuales o en aplicaciones de realidad aumentada. Al utilizar un sistema de estimación de pose, es posible crear interacciones efectivas en ambientes virtuales tomando en cuenta la posición de la cámara del dispositivo. El desarrollo de sistemas robustos de estimación de pose permite mejorar la experiencia del usuario en estas y otras aplicaciones [1].

Existen diferentes aproximaciones al problema de estimación de pose. Algunos métodos requieren varias imágenes de la misma escena o mediciones de profundidad provenientes de uno o más sensores. En este trabajo nos enfocamos en la estimación de pose monocular, que consiste en realizar el proceso de estimación a partir de una sola imagen. Esto es un problema retador, ya que al no contar con múltiples imágenes de la misma escena o información de profundidad, no es posible utilizar métodos convencionales de geometría multivista para obtener una estimación de la profundidad en la imagen y por ello no se puede desambiguar la escala tan fácilmente.

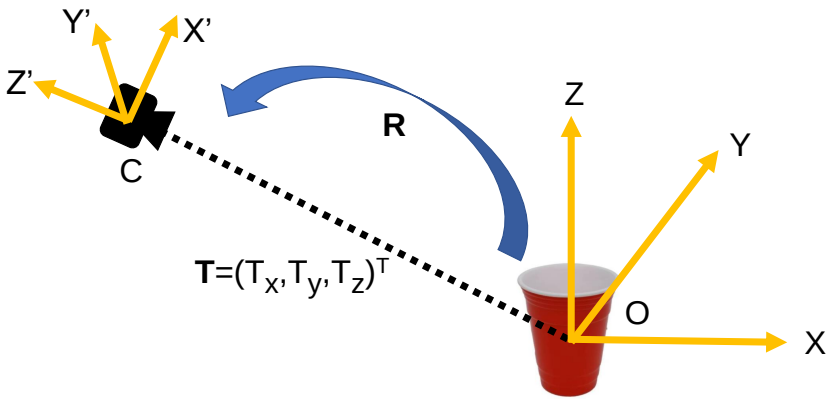

Figura 1.1: Estimación de pose como la obtención de una transformación geométrica tridimensional. En la figura $\boldsymbol{R}$ representa la matriz de rotación y $\boldsymbol{T}$ representa el vector de traslación.

Al contar únicamente con una imagen, es necesario para algunos de los sistemas de estimación de pose monocular tener una representación tridimensional del objeto de interés. Esta representación nos permite reconocer si el objeto de interés se encuentra en la escena al comparar las características presentes en la imagen y aquellas presentes en el modelo. Al analizar la presencia y distribución de estas características en la escena, podemos saber en qué parte de la imagen se encuentra el objeto, y conociendo los parámetros intrínsecos de la cámara, también podemos calcular su traslación y su rotación respecto a la cámara.

A la relación entre las características presentes tanto en la escena como en el modelo tridimensional se les conoce como correspondencias. Estas correspondencias sirven como entrada para el problema *Perspective-n-Point* (PnP), el cual a grandes rasgos consiste en obtener, a partir de las correspondencias, una matriz de transformación que maximiza la probabilidad de las características observadas. Es decir, a partir de una serie de correspondencias entre puntos en el modelo tridimensional y sus proyecciones en una imagen, permite calcular la pose del objeto. Dado que las correspondencias frecuentemente contienen ruido, este proceso por lo general se realiza en conjunto con métodos de estimación robusta como *Random Sample Consensus* (RANSAC) [7], el cual es capaz de generar hipótesis robustas ante datos atípicos.

El proceso descrito anteriormente para estimar la pose corresponde a una clase de métodos descritos en la literatura y puede descomponerse en tres subproblemas principales, como se muestra en la Figura 1.2. En general, cada una de las cláses de métodos propuestos hasta ahora tiene sus limitaciones y ha sido producto de décadas de investigación científica. La estimación de pose sigue siendo un problema abierto a pesar de haber múltiples propuestas de solución, esto se debe entre otras cosas a que no hay un solo sistema que funcione bien bajo todos los escenarios posibles. Las oclusiones, los cambios en la iluminación o la complejidad geométrica de los objetos de interés son algunos de los problemas que afectan de manera importante el rendimiento de los sistemas que ya han sido propuestos.

De forma general, podemos clasificar los sistemas de estimación de pose en métodos

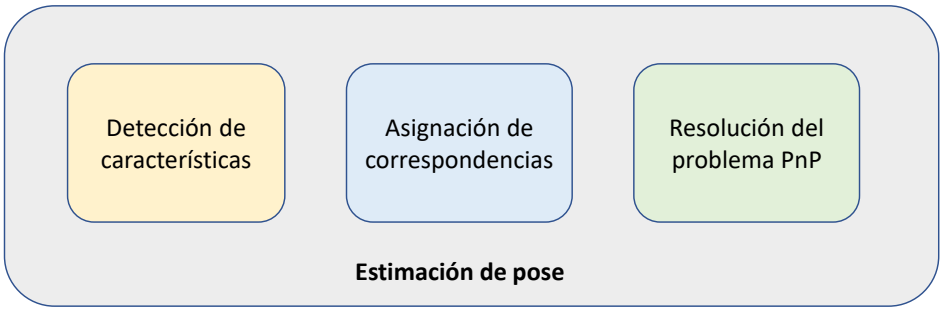

Figura 1.2: Subproblemas de la estimación de pose.

basados en características locales diseñadas a mano y aquellos basados en aprendizaje profundo. Los métodos basados en características locales diseñadas a mano utilizan la distribución geométrica de descriptores como SIFT [14], SURF [2] u ORB [18] como base para la detección de objetos en la escena [5]. Los métodos basados en aprendizaje profundo utilizan redes neuronales como base para la detección de características en la imagen. En particular, métodos basados en redes neuronales convolucionales, como PoseCNN [23], ConvPoseCNN [3] y DPOD [24], han demostrado un gran desempeño sobre el conjunto de datos LINEMOD [10].

A pesar de que los sistemas de estimación de pose basados en redes neuronales que realizan aprendizaje de principio a fin han demostrado tener la capacidad de resolver la tarea, ninguno de ellos resuelve de forma explícita el problema *Perspective-n-Point*. Esto se debe principalmente a que no es diferenciable.

1.1. Objetivo general

Crear un sistema de estimación de pose 6D de objetos monocular basado en redes neuronales convolucionales que pueda realizar aprendizaje de principio a fin. Dicho sistema debe obtener una imagen como entrada y ser capaz de detectar la presencia del objeto de interés, así como su traslación y rotación respecto de la cámara. Suponemos que los parámetros intrínsecos de la cámara son conocidos y que se cuenta con un modelo tridimensional del objeto de interés.

1.2. Metas y objetivos específicos

- Desarrollar una arquitectura que permita aprender la tarea de estimación de pose 6D de principio a fin y que resuelva el problema *Perspective-n-Point* de forma explícita.
- Analizar el rendimiento del sistema de estimación de pose propuesto, así como sus ventajas y desventajas.

1.3. Hipótesis

La hipótesis de este trabajo radica en que es posible aprender de principio a fin la estimación de pose 6D codificando el conocimiento de una aproximación PnP.

Al realizar aprendizaje de principio a fin utilizando una versión diferenciable de PnP, es posible propagar los gradientes desde la pose hasta la detección de las correspondencias y así mejorar el desempeño del sistema de estimación de pose en comparación con redes que realizan regresión de características a la pose como PoseCNN [23] y YOLO6D [22].

1.4. Contribución

Este trabajo demuestra que se puede integrar PnP a un sistema de estimación de pose basado en aprendizaje profundo. A diferencia de redes que realizan regresión de características a la pose, se utiliza una versión diferenciable de PnP. Esto constituye una aportación al área de estimación de pose ya que se ha sugerido que estas redes aprenden un conjuto de poses a partir del las imágenes de entrenamiento e intrínsecamente no están realizando la tarea de estimación de pose [4], por lo que su capacidad de generalización se ve reducida. Siguiendo esta lógica, la red neuronal propuesta debe tener un incremento en el desempeño en comparación con arquitecturas como PoseCNN [23] o YOLO6D [22].

Una de las ventajas de la arquitectura propuesta es que permite propagar los gradientes de los pesos desde la pose hasta las características de los objetos. Esto es algo deseable, ya que algunas de las arquitecturas del estado del arte requieren de un

proceso muy sofisticado para la preparación de los datos, lo que demanda una cantidad considerable de tiempo.

Otra ventaja de la arquitectura propuesta es que permite que el entrenamiento se haga tanto en datos reales como en datos sintéticos. Esto es muy útil cuando no se cuenta con suficientes imágenes del objeto en cuestión y únicamente se tiene el modelo tridimensional. Con esta información se pueden utilizar técnicas de aumentado de datos para generar un conjunto de datos de entrenamiento.

1.5. Resumen del contenido de la tesis

El resto de la tesis se encuentra organizada de la siguiente manera. En el capítulo 2 se describen las diferentes propuestas que se han hecho para resolver el problema de estimación de pose, así como las ventajas y desventajas de cada una. También se localiza la arquitectura propuesta dentro de dicha organización y se mencionan las diferencias con las propuestas existentes.

En el capítulo 3 se describen todos los fundamentos teóricos necesarios para comprender la arquitectura propuesta, como el funcionamiento de las redes convolucionales y el problema *Perspective-n-Point*.

En el capítulo 4, se presenta formalmente la propuesta y la metodología seguida en el proceso. Se justifica la arquitectura y se describen los datos utilizados, el preprocesamiento de los datos, las técnicas empleadas para la generación de modelos propios y los detalles de entrenamiento.

En el capítulo 5 se presentan las métricas de evaluación, así como la evaluación cualitativa y cuantitativa del modelo propuesto. Se compara el rendimiento contra los métodos pertenecientes al estado del arte en estimación de pose 6D bajo la métrica distancia promedio (ADD), tanto en datos reales como sintéticos.

Finalmente, en el capítulo 6 se presentan las conclusiones y el trabajo a futuro. Se discuten la metodología seguida, los resultados y la validación de la hipótesis.

Capítulo 2

Estado del arte

En este capítulo se aborda el trabajo que se ha realizado respecto a estimación de pose 6d monocular y que es relevante para la tesis. Además se posiciona la arquitectura propuesta dentro del estado del arte.

2.1. Clasificación de métodos de estimación de pose

Como se mencionó en el capítulo anterior, podemos dividir los métodos de estimación de pose en dos grandes grupos. El primer grupo consiste en aquellos métodos que ocupan características locales diseñadas por humanos, obtenidas mediante SIFT [14], SURF [2] u ORB [18]. Estos algoritmos detectan y describen características locales en las imágenes mediante laplacianos o diferencias de gaussianas. Su uso nos permite identificar objetos dentro de una imagen comparando los vectores de características del modelo del objeto con los de la escena.

El segundo grupo está conformado por aquellos métodos basados en aprendizaje profundo, especialmente en redes neuronales convolucionales. Este es otro paradigma para la detección de características. La red convolucional está conformada por una serie de capas que realizan operaciones como convolución y submuestreo. Las capas tienen parámetros entrenables que permiten el aprendizaje de la tarea en cuestión.

Dentro de los trabajos más recientes que utilizan descriptores tradicionales, podemos resaltar *Multiple Object Pose Estimation and Detection* (MOPED) [5], que presenta un marco de trabajo para la estimación de pose de múltiples objetos simultáneamente, con funcionamiento robusto en escenas complejas y un tiempo de respuesta pensado para aplicaciones en tiempo real. En la Figura 1.3 se puede observar el flujo de datos de MOPED.

El algoritmo central de MOPED es *Iterative Clustering-Estimation* (ICE), un algoritmo que realiza sucesivamente el agrupamiento de características y la estimación robusta de la pose del objeto. Al realizar el agrupamiento, se generan diferentes hipótesis de posibles objetos en la escena. Estas hipótesis a su vez se utilizan para refinar los grupos de características y ambos pasos iteran hasta converger. ICE está pensado de

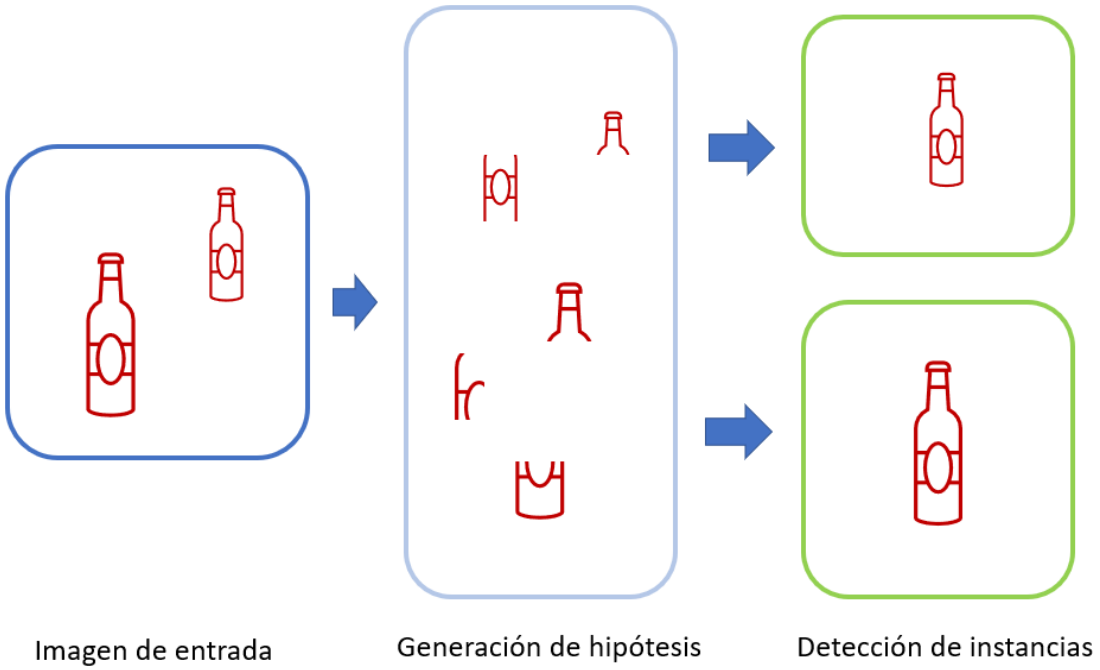

Figura 2.1: Flujo de datos en MOPED.

tal forma que puede implementarse en plataformas de cómputo paralelo.

Dentro del algoritmo de ICE, el problema de correspondencias se refiere a cómo agrupar las características tales que pertenezcan al mismo objeto. Una vez agrupados, el problema de estimación de pose se refiere a la generación de poses de objetos que maximicen la probabilidad de las correspondencias encontradas.

Como cualquier otro método de estimación de pose, MOPED está sujeto a la detección de correspondencias erróneas, por lo que se utilizan técnicas robustas de estimación, tales como RANSAC [7]. La presencia de varias instancias del mismo objeto dificulta el procesamiento de las correspondencias, ya que incluso si todas las correspondencias son perfectas, necesitan ser asociadas a una instancia del objeto.

Una de las limitaciones de MOPED es que debido al tipo de algoritmos de extracción de características locales que emplea, requiere que el objeto tenga una textura rica para que se puedan encontrar correspondencias. Cuando el objeto no cuenta con dicha textura el rendimiento baja considerablemente, llegando al punto de no poder reconocerlo. Es por ello que en años más recientes se ha optado por utilizar redes neuronales convolucionales en lugar de algoritmos de extracción de características locales diseñados por humanos.

A diferencia de MOPED, la arquitectura propuesta nos permite reconocer objetos cuya textura visual es muy homogénea (por ejemplo, un objeto liso de un solo color). Al ocupar redes neuronales convolucionales, enriquecemos la capacidad de reconocimiento.

2.2. Métodos basados en aprendizaje profundo

Los métodos de estimación de pose basados en redes neuronales convolucionales tienen un paradigma distinto tanto en la detección de características como en la asignación de las correspondencias y en la estimación de la pose. Dependiendo del tipo de red neuronal, se puede realizar el aprendizaje de principio a fin o por partes. Una estimación de pose de principio a fin consiste en una red cuya entrada es únicamente la imagen RGB y su única salida es la pose. En general estas redes se entrenan de forma supervisada con datos etiquetados, esto es, con pares imagen-pose. Existen otras redes neuronales que resuelven subproblemas de la estimación de pose, como la detección de características, la asignación de correspondencias o la resolución del problema *Perspective-n-Point*.

A continuación se discuten algunas de las arquitecturas con mayor rendimiento sobre el conjunto de datos de LINEMOD [10].

2.2.1. Aprendizaje de principio a fin

Dentro de las arquitecturas que realizan aprendizaje de principio a fin destaca PoseCNN [23], que es una red neuronal convolucional para la estimación de pose en 6D. Su arquitectura se muestra en la Figura 2.2. PoseCNN estima la traslación de un objeto localizando su centro en la imagen y prediciendo la distancia a la que se encuentra de la cámara. La rotación 3D del objeto se estima haciendo una regresión de las características del objeto a una representación en cuaternión de la pose. PoseCNN también introduce una función de pérdida que permite aprender de forma más eficiente la pose de objetos simétricos.

La idea clave dentro de PoseCNN es descomponer la tarea de la estimación de pose en diferentes componentes. Específicamente, PoseCNN realiza tres tareas. Primero hace segmentación semántica, es decir, predice una clase de objeto para cada pixel en la imagen de entrada.

Posteriormente, se predice un campo vectorial que apunta hacia el centro de cada objeto en la escena. Los vectores de este campo votan sobre la localización del centro del objeto en la imagen. Este sistema de votación se puede aplicar tanto a objetos con textura como sin textura y es robusto a oclusiones, ya que la red se entrena para votar sobre el centro incluso cuando el objeto está ocluido.

Asumiendo que se conocen los parámetros intrínsecos de la cámara y la ubicación del centro en 2D y su distancia a la cámara en 3D, se calcula la traslación del objeto. Finalmente, la rotación se obtiene realizando regresión a partir de las características obtenidas que se encuentren dentro de un *bounding box*, con lo cual se obtiene una estimación de la pose en una representación de cuaternión.

Dentro del artículo de PoseCNN se le presta especial atención al manejo de objetos simétricos, ya que diferentes orientaciones pueden generar observaciones idénticas. Al ignorar las simetrías como en otras aproximaciones la red recibe señales de pérdida inconsistentes. Motivados por esta observación, los autores desarrollaron una nueva

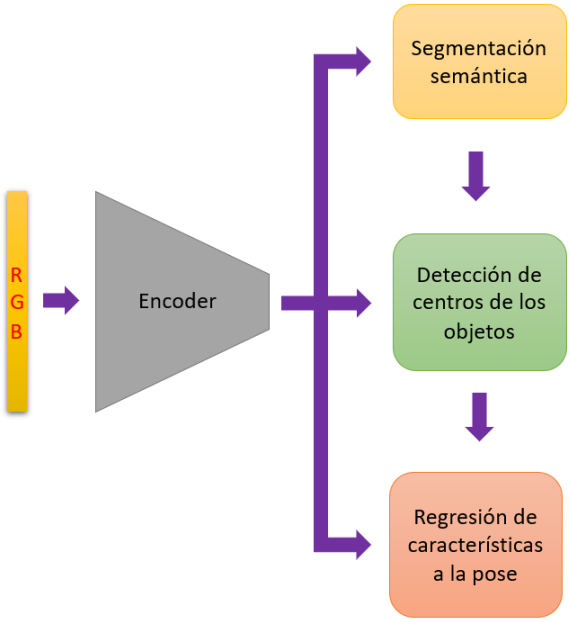

Figura 2.2: Arquitectura de PoseCNN.

función de pérdida que puede manejar simetrías.

Un trabajo posterior a PoseCNN, ConvPoseCNN [3], reduce el número de parámetros entrenables y el tiempo de entrenamiento sin sacrificar el desempeño.

Una diferencia muy importante entre la arquitectura que se propone en esta tesis y arquitecturas como PoseCNN o ConvPoseCNN es que estas redes no resuelven explícitamente el problema *Perspective-n-Point*, únicamente realizan una regresión de las características obtenidas de las capas convolucionales a una pose. Si bien esto ha demostrado tener un rendimiento superior a otras aproximaciones, al no resolver el problema PnP lo que se está haciendo en realidad es un mapeo de características a pose, con lo cual se pierde el conocimiento codificado en el problema PnP y por lo cual la capacidad de generalización es limitada [4].

Dentro de los sistemas más eficientes para la estimación de pose en el conjunto de datos LINEMOD se encuentra DPOD [24], cuya arquitectura se puede ver en la Figura 2.3. El sistema consiste en un bloque de correspondencias y otro de estimación de pose. El primer bloque contiene un codificador formado por capas convolucionales que extraen características de la imagen RGB de entrada. A la salida del codificador se encuentran tres decodificadores, que se encargan de obtener el canal U, el canal V y el ID de la clase. Los canales U y V codifican las correspondencias densas entre las características de la imagen y los modelos tridimensionales correspondientes. El ID se refiere a la segmentación semántica, que indica a qué objeto pertenece cada píxel de la imagen. Las salidas de los tres decodificadores son la entrada al bloque de estimación de pose. Los mapas UV obtenidos al concatenar los canales U y V, en conjunto con la segmentación semántica, se procesan para obtener las correspondencias con los modelos tridimensionales. Una vez que contamos con las correspondencias entre píxeles en la

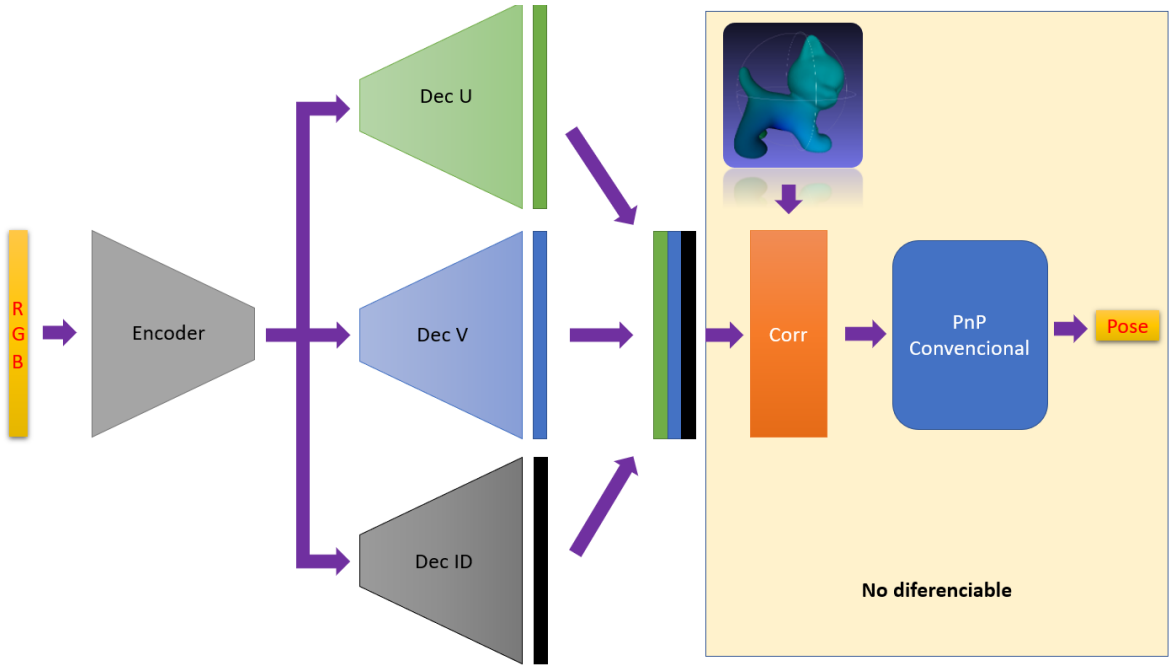

Figura 2.3: Arquitectura de DPOD.

imagen de entrada y vértices de los modelos tridimensionales, podemos utilizar algún solucionador PnP para estimar la pose. Dado que las correspondencias contienen ruido, se utiliza RANSAC [7] para generar una hipótesis de la pose robusta ante datos atípicos. El resultado es la pose codificada en una matriz de rotación y traslación.

Además de la primera estimación, DPOD tiene un sistema de refinamiento de pose que se entrena a partir de datos reales y sintéticos y recibe como entrada la imagen RGB y la pose estimada por el bloque de pose. A partir de esta estimación se genera una imagen artificial proyectando el modelo tridimensional con la pose estimada. Ambas imágenes entran a la red de refinamiento, cada una a un codificador que extrae un vector de características para cada imagen. Estos vectores son restados y el vector resultante entra a un nuevo codificador. El vector resultante alimenta a tres decodificadores que se encargan cada uno de estimar una componente de la pose, tanto un vector de traslación como la matriz de rotación.

A diferencia de DPOD, la arquitectura propuesta presenta una solución de principio a fin. Si bien es cierto que el rendimiento de DPOD es superior al de muchas otras arquitecturas sobre LINEMOD, la red únicamente realiza la estimación de la segmentación semántica y de los mapas de correspondencias. El problema PnP es resuelto de manera convencional después de que se han obtenido las correspondencias. Esto implica que el proceso de estimación de pose mediante PnP no beneficia en absoluto el entrenamiento de la red mediante la actualización de los pesos y sesgos. La arquitectura propuesta permite que la información de la pose influya directamente en el entrenamiento de las capas convolucionales de la red, con lo cual podríamos mejorar el rendimiento de la red, al utilizar el conocimiento codificado en el problema *Perspective-n-Point*.

Otro sistema que pertenece al estado del arte en estimación de pose 6D es *Pixel-*

wise Voting Network (PVNet) [16], que realiza el proceso mediante dos etapas. Su arquitectura se puede observar en la Figura 2.4. En la primera, utiliza una red neuronal convolucional que extrae características de la imagen de entrada con una arquitectura tipo codificador-decodificador. La salida de esta red es la segmentación semántica, que indica a qué clase pertenece cada píxel, además de un tensor que codifica un campo vectorial que apunta hacia cada uno de los puntos clave del objeto. Desde luego, dado que la nube de puntos del modelo tridimensional tiene demasiados vértices, se realiza una selección de puntos mediante el algoritmo *Farthest Point Sampling* (FPS). Este algoritmo se encarga de realizar una selección de puntos sobre la superficie del modelo de tal forma que estén lo más alejados posibles entre sí.

La salida de la red convolucional proporciona una distribución de probabilidad sobre dónde se encuentran los puntos clave del objeto. Esto hace que el sistema sea robusto a oclusiones, dado que aunque los puntos clave no estén presentes en la escena, siempre se tiene una predicción de la dirección en la cual están. Dado que el manejo de correspondencias se hace de forma probabilística, los autores de PVNet además proponen una versión de PnP basada en la incertidumbre.

El sistema basado en votos obtiene una distribución de probabilidad para cada punto clave del objeto. Dada la media y la matriz de covarianza se calcula la pose 6D minimizando la distancia de Mahalanobis:

$$\min_{\boldsymbol{R},\boldsymbol{t}} \sum_{k=1}^{K} (\tilde{x}_k - \mu_k)^T \Sigma_k^{-1} (\tilde{x}_k - \mu_k)$$

$$\tilde{x}_k = \pi(\boldsymbol{R} X_k + \boldsymbol{t})$$

donde X_k es la coordenada en 3D del punto clave, $\tilde{x_k}$ es la proyección 2D de X_k y π es la función de proyección en perspectiva. El problema de minimización se hace mediante el algoritmo Levenberg-Marquardt.

A diferencia del sistema multietapa de PVNet, el sistema de estimación de pose propuesto en esta tesis es capaz de aprender el proceso de principio a fin. Esto se logra debido a que tanto la versión de PnP como la asignación de correspondencias a partir de las características son completamente diferenciables.

Otro sistema de estimación de pose del estado del arte es SSD-6D [11], cuya arquitectura mostrada en la Figura 2.5 se basa en un esquema totalmente convolucional en el cual se obtienen seis mapas de características a diferentes escalas siguiendo una arquitectura tipo InceptionV4 [21]. Al realizar la convolución de cada capa con filtros dedicados a determinar la clase del objeto, se obtiene un *bounding box* y diferentes puntuaciones para posibles puntos de vista y rotaciones, a partir de las cuales se construye una hipótesis de la pose.

Algunas de las características que resaltan de esta arquitectura es que está entrenada por completo con datos sintéticos mediante técnicas de proyección. Se toma el modelo tridimensional del objeto y se realiza una proyección utilizando los parámetros intrínsecos de la cámara. El parche de color obtenido se coloca en un fondo de MS-COCO [13]. De

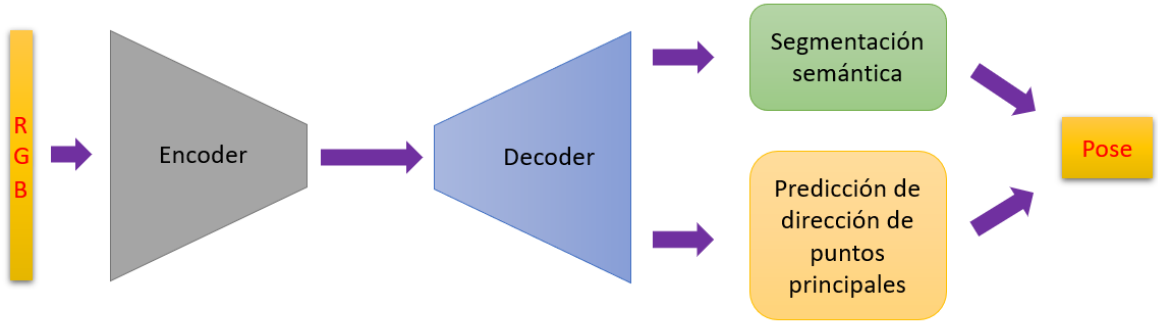

Figura 2.4: Diagrama de la arquitectura de pixel-wise Voting Network

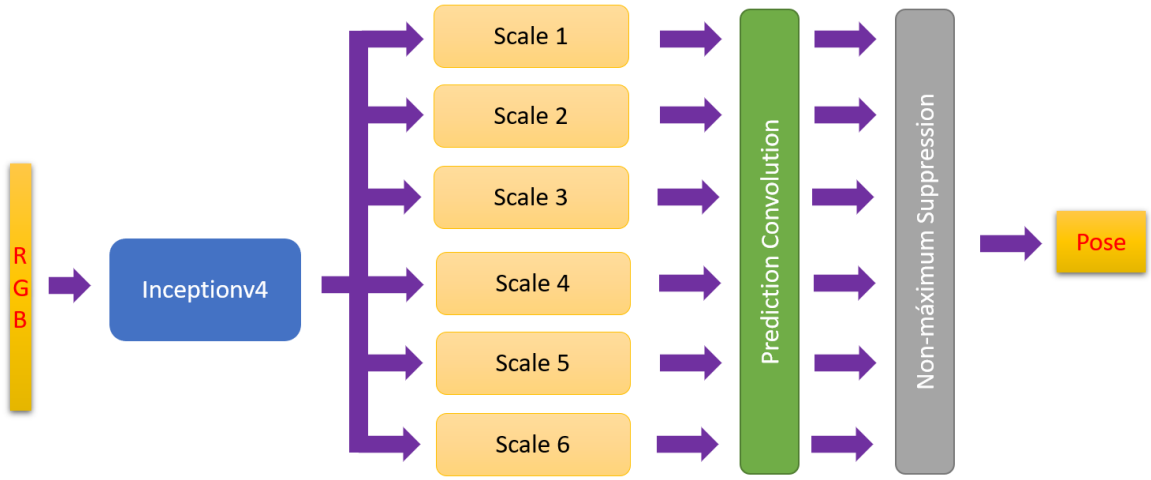

Figura 2.5: Diagrama general del funcionamiento de SSD-6D

esta forma se puede generar un conjunto de datos de entrenamiento tan grande como se requiera.

Al convertir el problema de estimación de pose en un problema de clasificación de puntos de vista discretos, se pierde un poco la precisión y se vuelve propenso a errores debido a las simetrías del objeto. La ventaja es que al limitar el espacio de búsqueda para la red, su tiempo de inferencia es bajo comparado con otras redes utilizadas para la estimación de pose.

A diferencia de SSD-6D la red propuesta en este trabajo puede entrenarse tanto con datos sintéticos como con datos reales y utiliza una aproximación PnP para la resolución de la estimación de pose, con lo cual se aprovecha el conocimiento geométrico que se tiene acerca del problema.

Finalmente dentro de las arquitecturas con aprendizaje de principio a fin, YOLO6D [22] (Figura 2.6) consta de una red neuronal convolucional tipo codificador, en la cual

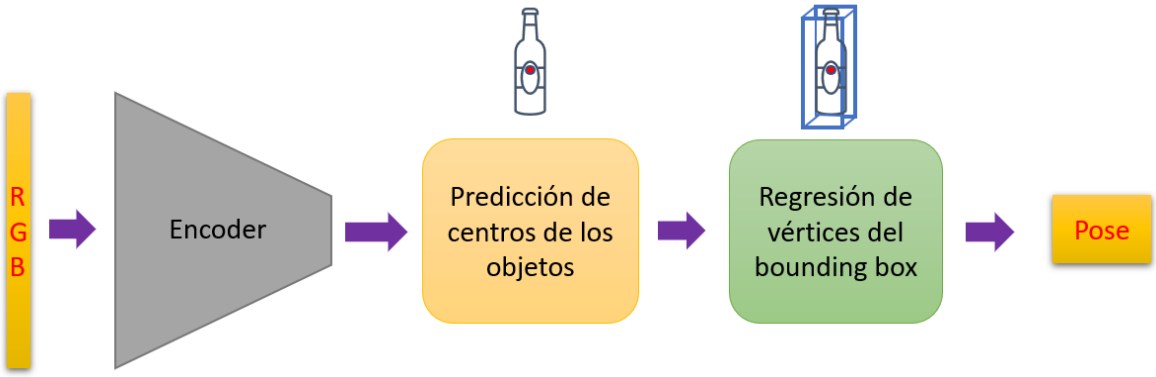

Figura 2.6: Diagrama general de la arquitectura de YOLO6D.

se extraen características de la imagen de entrada y se hace regresión para obtener la posición de 8 puntos clave asociados a cada objeto. Estos ocho puntos corresponden a las proyecciones de un bounding box tridimensional que contiene al objeto en la imagen.

Durante el proceso, se divide el área de la imagen en una malla cuadriculada. Para cada celda se obtiene una predicción de la localización de los puntos bidimensionales correspondientes a los 8 puntos clave, que son los vértices de un prisma que contiene al objeto. Una vez que el sistema tiene la posición de los puntos en la imagen, el sistema ocupa un solucionador PnP para poder resolver el problema de estimación de pose.

En YOLO6D, no se utilizan los puntos correspondientes al modelo tridimensional del objeto sino a puntos externos a él, lo cual afecta la calidad de las correspondencias y la precisión en la estimación de la pose. Además, al realizar la regresión de los vértices del *bounding box*, el número de correspondencias está restringido. A comparación de este método, la arquitectura propuesta realiza la estimación de la pose mediante PnP dentro de la red, por lo cual el aprendizaje se hace de principio a fin.

2.2.2. Aprendizaje de subtareas de estimación de pose

Además de los sistemas basados en redes neuronales convolucionales que hacen aprendizaje de principio a fin para la estimación de pose, también existen propuestas que se encargan de resolver las subtareas de forma independiente, como la detección de características y la asignación de correspondencias entre las mismas, seguidos de un procesamiento para poder realizar la estimación de la pose. Entre estos sistemas destacan SuperPoint [6] y SuperGlue [19].

SuperPoint es un marco de trabajo para detección de puntos de interés y descriptores. Su modelo totalmente convolucional opera en imágenes completas y calcula puntos y descriptores a nivel píxel. El proceso que realiza es análogo al realizado por SIFT [14]. A partir de ambos algoritmos se pueden obtener la posición y los descriptores de las características en la imagen.

SuperPoint tiene varias etapas de entrenamiento, mostradas en la Figura 2.7. Primero

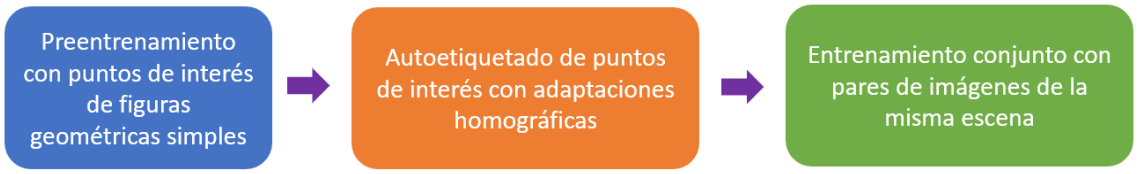

Figura 2.7: Diagrama general del entrenamiento de SuperPoint.

se entrena la red convolucional sobre millones de ejemplos de un conjunto de datos sintético, que consiste en figuras geométricas simples, donde no hay ambigüedad en los puntos de interés. Con esto se obtiene un detector de puntos. A partir de la red obtenida se hace un proceso de autoetiquetado con nuevas imágenes. Finalmente se puede hacer un entrenamiento en conjunto de una red neuronal totalmente convolucional que extrae puntos de interés y descriptores a partir de imágenes.

Por su parte, SuperGlue es una red neuronal que se encarga de encontrar correspondencias. Estas correspondencias están representadas mediante grafos con pesos variables. Podemos ver este proceso como uno análogo a los algoritmos utilizados tradicionalmente para hacer asignación de correspondencias.

La red neuronal considera que se tienen dos imágenes A y B, cada una con un conjunto de características formadas por un conjunto de posiciones asociadas a descriptores visuales. Las posiciones además contienen un porcentaje de confianza. Una de las ventajas de este método es que puede ser utilizado tanto con descriptores tradicionales como también con arquitecturas como SuperPoint.

Las correspondencias son asignadas resolviendo un problema diferenciable de transporte óptimo, cuyos costos son predichos a partir de una red neuronal. Introducen un mecanismo flexible de agregación de contexto basado en atención, que permite a la red razonar acerca de la escena subyacente y las asignaciones de características. Comparado con heurísticas tradicionales diseñadas a mano, la técnica aprende primitivas sobre transformaciones geométricas y regularidades del mundo tridimensional a través de entrenamiento entre pares de imágenes. SuperGlue tiene un mejor desempeño sobre otros enfoques de aprendizaje y alcanza resultados que mejoran el estado del arte en la tarea de estimación de pose en ambientes retadores del mundo real.

A pesar de que SuperPoint tiene un excelente rendimiento para la detección de puntos característicos en espacios interiores como cuartos, resulta no ser tan efectivo para la detección de características en objetos pequeños. Esto se debe a que fue entrenado para poder identificar esquinas, que son características que raramente están presentes en objetos pequeños o sin textura. Por su parte, SuperGlue resuelve adecuadamente el problema de asignación de correspondencias, sin embargo, está pensado para procesar vectores de características locales diseñadas por humanos, por lo cual no es una opción adecuada al utilizar redes convolucionales de principio a fin.

Otra de las subtareas de estimación de pose consiste en relacionar las características con los modelos tridimensionales de los objetos para obtener las correspondencias. Para

poder hacer este emparejamiento, se realizan operaciones de indizado que por lo general no son diferenciables. En arquitecturas que realizan aprendizaje de principio a fin, es necesario que este proceso sea diferenciable para poder propagar los gradientes de los pesos y sesgos de la red. Para lograrlo, existen algunos mecanismos como el cálculo de un promedio ponderado, aunque existen mejores alternativas como DSNT.

Differentiable Spatial to Numerical Transform [15] está diseñado justamente para poder realizar la regresión de coordenadas numéricas que correspondan a alguna posición de interés dentro de una imagen. La incorporación de DSNT a la arquitectura propuesta permite incorporar de forma natural las coordenadas bidimensionales de las correspondencias a BPnP como se explicará más adelante.

Capítulo 3

Marco teórico

En este capítulo se abordan los fundamentos teóricos necesarios para comprender la arquitectura propuesta. Se presenta a detalle el problema *Perspective-n-Point* y las diferentes soluciones que se han propuesto. También se abordan las redes neuronales convolucionales y las redes de refinamiento para estimación de pose 6d.

3.1. Estimación de pose

Como se mencionó anteriormente, la estimación de pose es un problema central en visión computacional y ha sido estudiado por décadas. Una de las primeras aproximaciones para obtener la pose de un objeto dada una serie de correspondencias entre puntos en tres dimensiones y sus proyecciones en el plano de la imagen fue el problema *Perspective-3-Point* (P3P) utilizando una cámara estenopeica (Figura 3.1). En esta aproximación se usan tan solo tres correspondencias y existen varios métodos para poder resolverlo. El más eficiente es *Lambda Twist* [17], que explota las ecuaciones elípticas subyacentes y resuelve el problema mediante diagonalización. Una característica clave de este método es que no calcula soluciones duplicadas o inválidas.

La generalización del problema P3P a n puntos se conoce como *Perspective-n-Point* problem (PnP) y existen varias soluciones propuestas en la literatura. Uno de los métodos más conocidos para resolver este es *Direct Linear Transformation* (DLT). Dicho método se basa en una serie de ecuaciones de similitud entre las correspondencias.

Durante el proceso de estimación de pose, existe un conjunto de correspondencias entre puntos 2D en la imagen y puntos 3D en el modelo del objeto. La estimación de pose consiste en encontrar la matriz de proyección $\boldsymbol{P}$ tal que explique la proyección de los puntos al plano de la imagen.

$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}],$$

donde $\boldsymbol{R}$ y $\boldsymbol{t}$ son la matriz de rotación y el vector de traslación del objeto respecto a la cámara respectivamente y $\boldsymbol{K}$ es la matriz de parámetros intrínsecos de la cámara, dada por

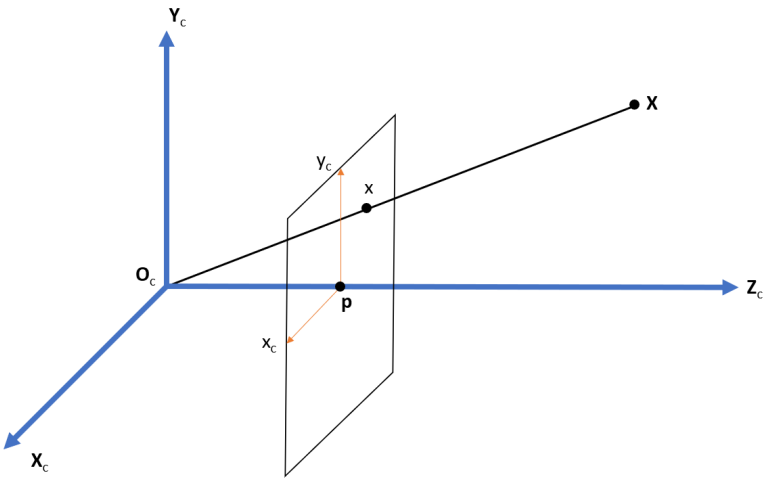

Figura 3.1: Descripción geométrica del modelo de cámara estenopeica.

$$\boldsymbol{K} = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Los parámetros f_x y f_y describen la distancia focal entre el plano de la imagen y el plano paralelo que pasa por la cámara. Por su parte, los parámetros x_0 y y_0 corresponden al desplazamiento existente entre el eje principal de la cámara y el origen de la imagen. El parámetro s se refiere a la distorsión por cizallamiento.

Una vez que conocemos la matriz de parámetros intrínsecos y las correspondencias, podemos estimar la matriz $[\boldsymbol{R}|\boldsymbol{t}]$ mediante la resolución de un sistema de ecuaciones matriciales. La matriz de rotación se construye a partir del producto de matrices elementales de rotación sobre los ejes principales de un sistema de coordenadas tridimensional. Para realizar rotaciones a partir de cada uno de los ejes coordenados, se determina un ángulo θ y se construyen las matrices de la siguiente forma

$$\boldsymbol{R}_x(\theta_1) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_1) & -\sin(\theta_1) \\ 0 & \sin(\theta_1) & \cos(\theta_1) \end{pmatrix},$$

$$\boldsymbol{R}_y(\theta_2) = \begin{pmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) \\ 0 & 1 & 0 \\ -\sin(\theta_2) & 0 & \cos(\theta_2) \end{pmatrix},$$

$$\boldsymbol{R}_z(\theta_3) = \begin{pmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Para construir una matriz correspondiente a una rotación arbitraria dados los ángulos de Euler $(\theta_1, \theta_2, \theta_3)$ simplemente multiplicamos las matrices

$$\boldsymbol{R} = \boldsymbol{R}_x(\theta_1)\boldsymbol{R}_y(\theta_2)\boldsymbol{R}_z(\theta_3).$$

El vector de traslación codifica el desplazamiento respecto al origen en dirección de cada uno de los ejes coordenados.

Lo que hace diferente al sistema de ecuaciones obtenido en DLT es que las ecuaciones están sujetas a un factor de proporcionalidad, por lo que no se puede hacer el cálculo de forma trivial. Para poder calcular las 12 variables correspondientes a los valores de la matriz $\boldsymbol{P}$, es necesario tener al menos 6 correspondencias. Sin embargo, en caso de que se conozcan los parámetros intrínsecos de la cámara, es posible calcular $\boldsymbol{P}$ con sólo tres correspondencias.

Adicionalmente a DLT, existen algoritmos iterativos, que buscan estimar la pose al minimizar el error de reproyección utilizando mínimos cuadrados no lineales. El error de reproyección se calcula como el cuadrado de la diferencia entre la proyección del punto tridimensional dada la matriz $\boldsymbol{P}$ y la posición real del punto en la imagen. Especificamente, la función para proyectar un punto $\mathbf{p}^{(i)} \in \mathbb{R}^3$ al plano de la imagen está dada por

$$\mathbf{x}^{(i)} = f(\mathbf{p}^{(i)}; \boldsymbol{R}, \boldsymbol{t}, \boldsymbol{K}), \tag{3.1}$$

donde $\mathbf{p}^{(i)}$ corresponde a las coordenadas 3D del punto y $\mathbf{x}^{(i)} \in \mathbb{R}^2$ corresponde a las coordenadas del punto proyectado. De este modo, obtenemos un vector residual, del cual podemos calcular derivadas parciales respecto a los parámetros de la cámara para corregir los parámetros iterativamente hasta minimizar el error.

Otra manera de minimizar el error de reproyección es construir la función de error al descomponer la ecuación 3.1 en operaciones más básicas. Primero se traslada el punto a la cámara, después se rota, luego se corrige la perspectiva y finalmente se aplica la matriz de parámetros intrínsecos de la cámara.

A pesar de que existen muchos métodos propuestos para resolver el problema de estimación de pose, sigue siendo retador debido a que ningún método funciona bajo condiciones arbitrarias en el mundo real. La gran cantidad de objetos, formas, apariencias, niveles de iluminación, entre otros factores, provocan que la dificultad del problema aumente considerablemente.

Con el surgimiento de cámaras de profundidad se han desarrollado métodos que permiten reconocer objetos sin textura a partir de información RGB-D [9]. También hay métodos que detectan objetos a partir de plantillas [8] o sistemas que realizan una regresión de píxeles a coordenadas de objetos 3D para poder establecer correspondencias [22]. En todos estos métodos las oclusiones y las simetrías afectan mucho el rendimiento.

3.2. *Perspective-n-Point*

Para resolver el problema PnP para una cantidad arbitraria de puntos, actualmente se tienen una serie de métodos que forman parte del estado del arte. Para la estimación

de pose 6D monocular, dependiendo de la cantidad de correspondencias que tengamos, así como de su calidad y de la capacidad de procesamiento con la que contemos, podemos decidir utilizar uno de los siguientes métodos: P3P, EPnP [12], RANSAC [7] o BPnP [4]. A continuación se describe cada uno de ellos.

3.2.1. P3P

Dada una cámara calibrada $\boldsymbol{C}$ y n correspondencias entre puntos tridimensionales $\boldsymbol{p}_i$ y sus imágenes $\boldsymbol{u}_i$, cada par de correspondencias $\boldsymbol{p}_i \leftrightarrow \boldsymbol{u}_i$ y $\boldsymbol{p}_j \leftrightarrow \boldsymbol{u}_j$ da una restricción sobre las distancias $\boldsymbol{x}_i = ||\boldsymbol{p}_i - \boldsymbol{c}||$ y $\boldsymbol{x}_j = ||\boldsymbol{p}_j - \boldsymbol{c}||$ entre la cámara y los puntos. Expresamos la distancia entre $\boldsymbol{p}_i$ y $\boldsymbol{p}_j$ como

$$d_{ij}^2 = x_i^2 + x_j^2 - 2x_i x_j \cos(\theta_{ij})$$

donde θ_{ij} es el ángulo entre el centro de la cámara y los puntos p_i y p_j. Podemos calcular $\cos(\theta_{ij})$ como

$$\cos(\theta_{ij}) = \frac{\mathbf{u}_i^T \boldsymbol{C} \mathbf{u}_i}{(\mathbf{u}_i^T \boldsymbol{C} \mathbf{u}_i)^{1/2} (\mathbf{u}_j^T \boldsymbol{C} \mathbf{u}_j)^{1/2}}$$

donde $\boldsymbol{C} = (\boldsymbol{K}\boldsymbol{K}^T)^{-1}$. La restricción al sistema se puede escribir como

$$f_{ij}(x_i, x_j) = x_i^2 + x_j^2 - 2x_i x_j \cos(\theta_{ij}) - d_{ij}^2 = 0$$

Para $n = 3$ se tiene

$$f_{12}(x_1, x_2) = 0$$
$$f_{13}(x_1, x_3) = 0$$
$$f_{23}(x_2, x_3) = 0$$

Utilizando eliminación de variables con $x = x_1^2$ obtenemos

$$g(x) = a_5 x^4 + a_4 x^3 + a_3 x^2 + a_2 x + a_1 = 0$$

que tiene a lo más cuatro soluciones. Para obtener una solución única tomamos en cuenta que $\mathbf{x}^{(i)}$ debe ser positiva y se puede desambiguar la solución utilizando un cuarto punto. Se resuelve para cada subconjunto de 3 puntos y se encuentra la solución común.

Cuando únicamente contamos con tres correspondencias, el problema se puede resolver de forma analítica, sin embargo en la práctica las correspondencias contienen ruido y esto nos lleva a un resultado incorrecto. Es por ello que es recomendable utilizar métodos que puedan generar hipótesis robustas a partir de conjuntos más grandes de correspondencias. En la Figura 3.2 se puede observar un diagrama del problema.

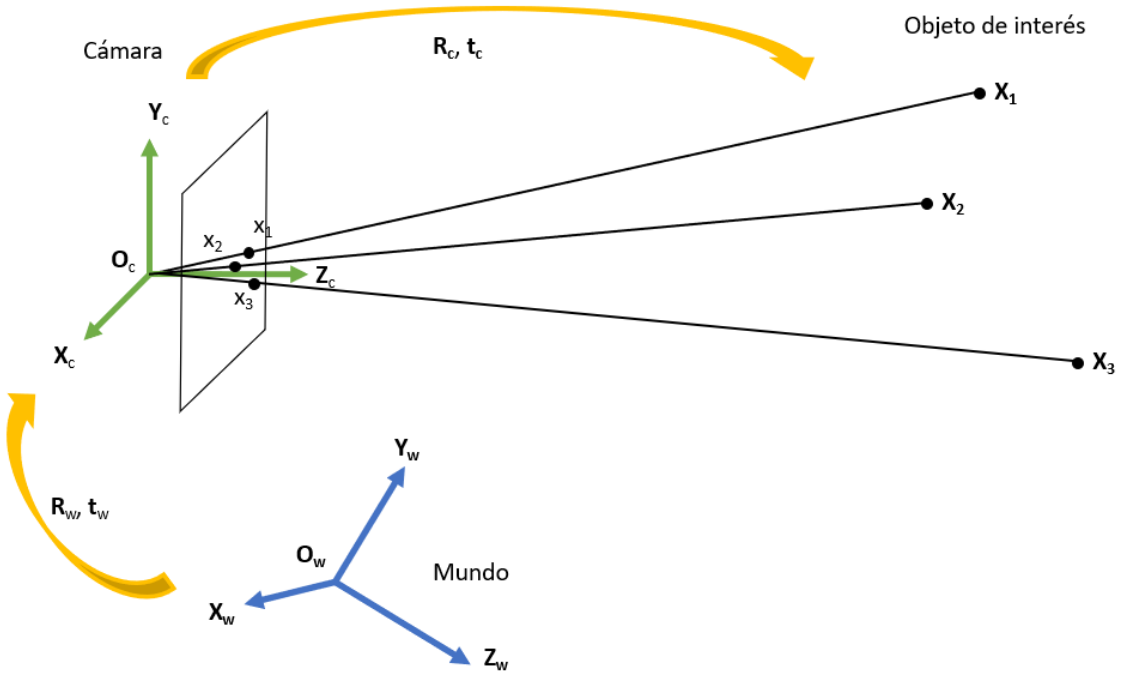

Figura 3.2: Perspective-3-Point problem.

3.2.2. *Efficient PnP* (EPnP)

Este método supone que se tiene un conjunto de n puntos de referencia cuyas coordenadas tridimensionales son conocidas en el sistema de referencia del mundo y cuyas proyecciones bidimensionales también son conocidas. La mayoría de los enfoques para resolver el sistema de estimación de pose buscan encontrar la profundidad de los puntos de referencia en el sistema de coordenadas de la cámara. Sin embargo, EPnP expresa las coordenadas de dichos puntos como una suma ponderada de cuatro puntos virtuales de control no coplanares. Con esta formulación las coordenadas de los puntos de control en el sistema de coordenadas de la cámara se convierten en la incógnita del problema. Para una cantidad grande de puntos, este es un número mucho menor de incógnitas que las de n profundidades, lo cual permite que la implementación sea mucho más eficiente.

La solución al problema se expresa como un vector que se encuentra en el centro de una matriz de tamaño $2n \times 12$ o $2n \times 9$. Esta matriz se expresa como $\boldsymbol{M}$ y puede ser calculada a partir de coordenadas tridimensionales de puntos de referencia y sus proyecciones bidimensionales en la imagen. De hecho, es una suma ponderada de todos los eigenvectores no nulos de $\boldsymbol{M}$. A partir del cálculo de la distancia entre cada uno de los puntos tridimensionales de interés y los puntos de control, se obtiene un sistema de ecuaciones cuadráticas que puede ser resuelto a un costo computacional muy bajo. De hecho para $n > 15$ la parte más costosa computacionalmente es calcular $\boldsymbol{M}^T\boldsymbol{M}$, que crece linealmente con n.

Efficient PnP es un método capaz de procesar cualquier número de correspondencias

mayor a cuatro en tiempo lineal respecto al número de correspondencias. Sin embargo, este método es afectado de forma importante por datos atípicos, que se presentan comúnmente en la práctica debido al ruido y las correspondencias erróneas. También se debe destacar que este algoritmo no es diferenciable, por lo cual no se puede integrar a las redes neuronales que realizan estimación de pose para aprendizaje de principio a fin [12].

3.2.3. RANSAC

RANSAC es un método iterativo que permite estimar los parámetros de un modelo. Se presupone que se tiene un conjunto de observaciones, las cuales se desea ajustar a un modelo. La diferencia principal de RANSAC en comparación con otro tipo de métodos, como mínimos cuadrados, es que considera que existen datos atípicos. El algoritmo de RANSAC se puede resumir en los siguientes pasos

1. Se selecciona de forma aleatoria un subconjunto mínimo de los datos que permita estimar los parámetros del modelo y se presupone que no contiene datos atípicos.
2. Se estiman los parámetros del modelo para el subconjunto de datos seleccionado.
3. Se comprueba si el modelo obtenido explica al resto de los datos mediante una métrica definida para el modelo específico y se clasifican los datos en miembros del conjunto de concenso (*inliers*) o datos atípicos (*outliers*).
4. Si el modelo es suficientemente bueno se vuelven a estimar los parámetros utilizando todos los miembros del conjunto de concenso.
5. Se repiten los pasos del 1 al 4 hasta que se cumpla un número de iteraciones o la métrica definida sobrepase un umbral establecido.
6. Se devuelve el modelo con el mejor desempeño bajo la métrica definida.

En el caso de la estimación de la pose, al tener múltiples correspondencias, es posible que sólo un subconjunto de ellas sean correctas y el resto sean datos atípicos. Para este caso, el modelo puede ser cualquier algoritmo de estimación de pose a partir de correspondencias 2D-3D como EPnP. RANSAC se encarga de analizar subconjuntos de correspondencias y calcular cuál es la mejor pose que se ajusta a esos datos. Posteriormente valida con el resto de las correspondencias qué tan buena es la estimación y repite este proceso hasta llegar al modelo con menor error de reproyección. Este método permite hacer mucho más robusta la estimación en escenarios reales sujetos a errores. En la Figura 3.3 se puede observar un ejemplo simple de uso de RANSAC.

3.2.4. BPnP: PnP diferenciable

Con la proliferación de métodos para la estimación de la pose basados en redes neuronales, como PoseCNN [23] y YOLO6D [22], se optó por alternativas diferenciables

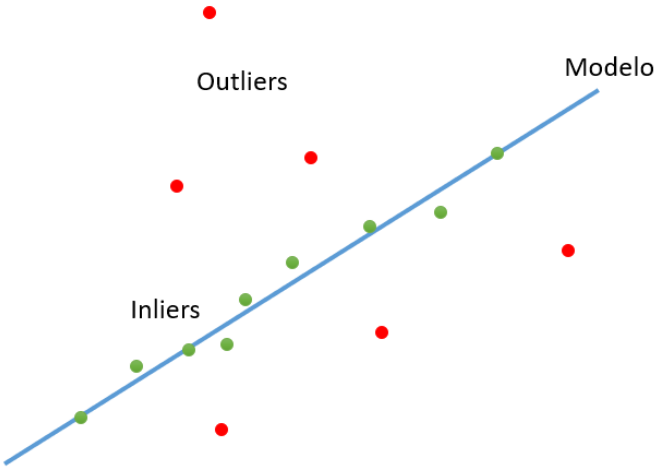

Figura 3.3: RANSAC

a la resolución del problema PnP. Usualmente estos métodos realizan una regresión de la pose o alguna de sus componentes a partir de las características obtenidas por la red convolucional. En algunos casos, en lugar de realizar la regresión de la pose se realiza la regresión de la posición de los vértices de un *bounding box* tridimensional que contiene al objeto. Para cualquiera de estos métodos, se evita resolver el problema PnP de forma explícita debido a que algoritmos como P3P o EPnP no son diferenciables.

Al realizar la estimación de pose mediante regresión, resolver PnP es sustituido por un mapeo. Sin embargo, al realizar este proceso, la información geométrica codificada en la resolución del problema *Perspective-n-Point* se pierde y la red no se ve beneficiada por el conocimiento acerca del problema, únicamente realiza un mapeo de las características de la imagen a alguna pose mediante los datos con los cuales fue entrenada [4].

BPnP [4] introduce una formulación diferenciable de PnP, con la cual es posible calcular los gradientes utilizando la diferenciación implícita. Al tener acceso a los gradientes es posible integrar BPnP con redes neuronales para realizar aprendizaje de principio a fin de estimación de pose.

La propuesta denota un solucionador PnP como una función

$$\boldsymbol{Y} = g(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{K})$$

donde $\boldsymbol{Y}$ es la pose con 6 grados de libertad de una cámara con parámetros intrínsecos denotados por la matriz $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ de n correspondencias 2D-3D

$$\boldsymbol{x} = \begin{bmatrix} \mathbf{x}_1^T & \boldsymbol{x}_2^T & \dots & \boldsymbol{x}_1^T \end{bmatrix} \in \mathbb{R}^{2n\times1}$$
$$\boldsymbol{z} = \begin{bmatrix} \mathbf{z}_1^T & \boldsymbol{z}_2^T & \dots & \boldsymbol{z}_1^T \end{bmatrix} \in \mathbb{R}^{3n\times1}$$

donde $(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)})$ representa la correspondencia entre el punto tridimensional $\boldsymbol{z}^{(i)}$ y el punto bidimensional $\boldsymbol{x}^{(i)}$.

Sea $\pi(\boldsymbol{z}^{(i)}|\boldsymbol{Y},\boldsymbol{K})$ una transformación proyectiva de un punto tridimensional al plano de la imagen utilizando la pose $\boldsymbol{Y}$ y la matriz de parámetros intrínsecos $\boldsymbol{K}$. La evaluación de la función g para un conjunto de correspondencias requiere resolver el problema de optimización dado por

$$\boldsymbol{y} = \arg\max_{\boldsymbol{y}\in SE(3)} \sum_{i=1}^{n} ||\boldsymbol{r}^{(i)}||_2^2, \tag{3.2}$$

$$\boldsymbol{r}^{(i)} = \pi(\boldsymbol{z}^{(i)}|\boldsymbol{Y},\boldsymbol{K}). \tag{3.3}$$

Utilizando el teorema de funciones implícitas se define una función de restricción $h(\mathbf{a},\mathbf{b})$. Para el problema de estimación de pose, dicha función se construye a partir de $\boldsymbol{x}$, $\boldsymbol{Y}$, $\boldsymbol{z}$ y $\boldsymbol{K}$. Dado que la función h es de dos variables, construimos $\mathbf{a}$ a partir de $\boldsymbol{x},\boldsymbol{z},\boldsymbol{K}$ y $\boldsymbol{b}$ con $\boldsymbol{Y}$.

Denotamos la función objetivo del solucionador PnP o como

$$o(\boldsymbol{x},\boldsymbol{Y},\boldsymbol{z},\boldsymbol{K}) = \sum_{i=1}^{n} ||\boldsymbol{r}_i||_2^2. \tag{3.4}$$

Dado que la pose $\boldsymbol{Y}$ es un óptimo local para la función objetivo, una restricción estacionaria se puede establecer tomando la primer derivada del objetivo con respecto a $\boldsymbol{Y}$

$$\frac{\partial o(\boldsymbol{x},\boldsymbol{Y},\boldsymbol{z},\boldsymbol{K})}{\partial \boldsymbol{Y}}\Big|_{\boldsymbol{y}=g(\boldsymbol{x},\boldsymbol{z},\boldsymbol{K})} = \mathbf{0}. \tag{3.5}$$

Así, la pose de salida del solucionador PnP

$$\boldsymbol{y} = \begin{bmatrix} h_1 & h_2 & \cdots & h_m \end{bmatrix}^T$$

donde

$$h_j = \frac{\partial o(\boldsymbol{x},\boldsymbol{Y},\boldsymbol{z},\boldsymbol{K})}{\partial \boldsymbol{Y}_j}, j = 1,\ldots,m \tag{3.6}$$

$$= 2\sum_{i=1}^{n} \langle \boldsymbol{r}_i, \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{Y}_j} \rangle \tag{3.7}$$

$$= \sum_{i=1}^{n} \langle \boldsymbol{r}_i, \boldsymbol{c}_{ij} \rangle \tag{3.8}$$

con

$$\boldsymbol{c}_{ij} = -2\frac{\partial \pi_i}{\partial \boldsymbol{y}_j} \tag{3.9}$$

Durante la retropropagación se construye el Jacobiano de g con respecto a cada una de las entradas

$$\frac{\partial g}{\partial \boldsymbol{x}} = -\left[\frac{\partial h}{\partial \boldsymbol{Y}}\right]^{-1}\left[\frac{\partial h}{\partial \boldsymbol{x}}\right] \tag{3.10}$$

$$\frac{\partial g}{\partial \boldsymbol{z}} = -\left[\frac{\partial h}{\partial \boldsymbol{Y}}\right]^{-1}\left[\frac{\partial h}{\partial \boldsymbol{z}}\right] \tag{3.11}$$

$$\frac{\partial g}{\partial \boldsymbol{K}} = -\left[\frac{\partial h}{\partial \boldsymbol{Y}}\right]^{-1}\left[\frac{\partial h}{\partial \boldsymbol{K}}\right] \tag{3.12}$$

Dado el gradiente de la pose $\nabla \boldsymbol{Y}$, BPnP calcula los gradientes de g respecto de $\boldsymbol{x}, \boldsymbol{z}$ y $\boldsymbol{K}$

$$\nabla \boldsymbol{x} = \left[\frac{\partial g}{\partial \boldsymbol{x}}\right]^{T} \nabla \boldsymbol{Y} \tag{3.13}$$

$$\nabla \boldsymbol{z} = \left[\frac{\partial g}{\partial \boldsymbol{z}}\right]^{T} \nabla \boldsymbol{Y} \tag{3.14}$$

$$\nabla \boldsymbol{K} = \left[\frac{\partial g}{\partial \boldsymbol{K}}\right]^{T} \nabla \boldsymbol{Y} \tag{3.15}$$

A partir de estos cálculos, es posible propagar los gradientes del módulo BPnP al resto de una red neuronal convolucional. Esto nos permite realizar aprendizaje de principio a fin para resolver el problema de estimación de pose siguiendo una aproximación PnP.

3.3. Redes neuronales convolucionales

La operación de convolución es sumamente importante dentro del área de procesamiento de imágenes. Esta operación se realiza entre imágenes y filtros, y dependiendo de qué filtro se utilice es posible producir diferentes resultados como detección de bordes o emborronamiento de la imagen.

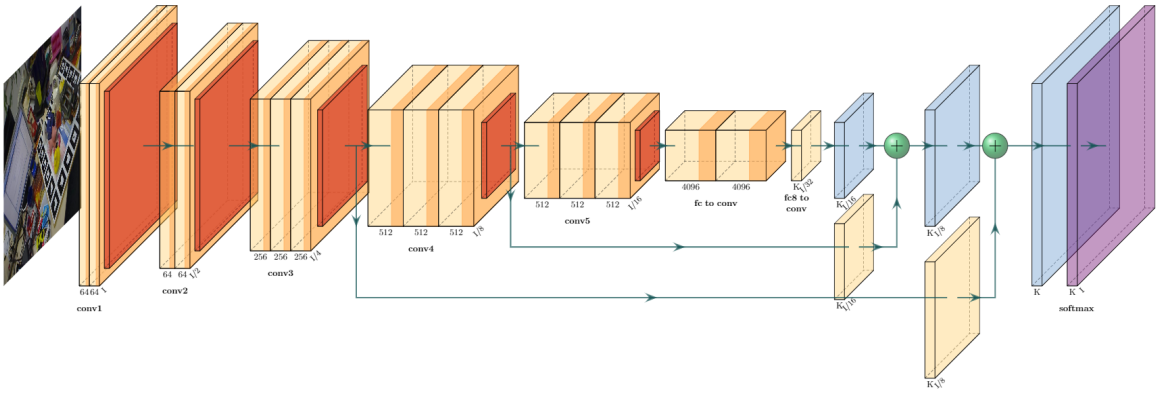

Figura 3.4: Estructura de una red neuronal convolucional. Figura dibujada utilizando *PlotNeuralNet*, disponible en *https://github.com/HarisIqbal88/PlotNeuralNet*.

Las redes neuronales convolucionales están basadas justamente en este principio. A diferencia del perceptrón multicapa, las redes neuronales convolucionales aprovechan la estructura espacial de las imágenes. La posición de un píxel RGB es importante dentro de la estructura de la imagen y está estrechamente relacionada con la composición de sus píxeles vecinos. En la Figura 3.4 podemos observar un ejemplo de arquitectura de una red neuronal convolucional.

Aprendiendo los filtros adecuados a partir de un conjunto de datos dado se pueden realizar tareas como clasificación de imágenes. A diferencia de los filtros diseñados por humanos, los pesos y sesgos que se obtienen al entrenar una red neuronal convolucional están optimizados para la tarea que queramos realizar. Un ejemplo clásico es la clasificación binaria de imágenes. Al darle una gran cantidad de ejemplos a la red neuronal, ésta optimiza sus filtros de tal forma que aprenda a realizar la clasificación.

Es por ello que las redes neuronales convolucionales han tenido un auge importante dentro del procesamiento de imágenes. Dentro del problema específico de estimación de pose, una de las ventajas importantes del uso de redes neuronales es la variedad de objetos que nos permite detectar. Al utilizar descriptores hechos por humanos como SIFT [14] u ORB [18] no podemos aprender a diferenciar objetos que cuentan con pocos descriptores, como lo son objetos de color homogéneo o con una textura pobre. Las redes neuronales convolucionales son capaces de diferenciar una gran variedad de objetos, desde aquellos con textura hasta aquellos sin textura o transparentes.

Una de las redes neuronales que hacen aprendizaje de principio a fin y que pertenecen al estado del arte sobre el conjunto de datos de LINEMOD [10] es DPOD [24]. La arquitectura desarrollada basa parte de su estructura en el funcionamiento de DPOD y es por ello que se explica a detalle en la siguiente sección.

3.4. Arquitectura de DPOD

Como se mencionó en el capítulo anterior, DPOD es una red neuronal que tiene como entrada una imagen RGB y como salida tres tensores. Su arquitectura se muestra

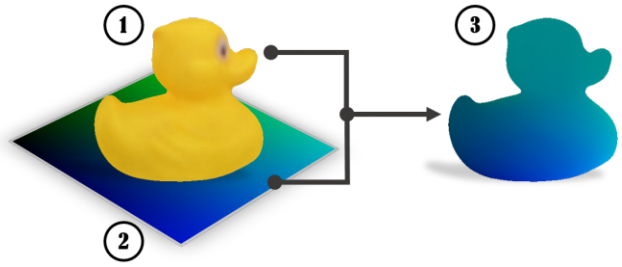

Figura 3.5: Modelo de correspondencias de DPOD. Figura tomada de [24].

en la Figura 3.6. El primer tensor tiene el ancho y alto de la imagen de entrada y codifica la segmentación semántica del objeto de interés. Para el caso de un solo objeto, cada píxel de la imagen nos indica si el objeto de interés está presente o no. El segundo tensor y el tercer tensor también tienen el ancho y alto de la imagen, y codifican los canales U y V de un mapa de correspondencias. Como se muestra en la figura 3.5 cada combinación de valores (U, V) es única y codifica a un vértice específico de la nube de puntos del objeto. Esto nos permite tener una relación biyectiva entre tonos de color UV y coordenadas XYZ del objeto.

El sistema consiste en un bloque de correspondencias y otro de estimación de pose. El primer bloque contiene un codificador formado por capas convolucionales que extraen características de la imagen RGB de entrada. A la salida del codificador se encuentran tres decodificadores, que se encargan de obtener el canal U, el canal V y el ID de la clase. Los canales U y V codifican las correspondencias densas entre las características de la imagen y los modelos tridimensionales correspondientes. El ID se refiere a qué objeto pertenece cada píxel de la imagen. Las salidas de los tres decodificadores son la entrada al bloque de estimación de pose. Los mapas UV que codifican las correspondencias, en conjunto con la segmentación semántica, se procesan para obtener la relación con los modelos tridimensionales. Una vez que contamos con las correspondencias entre píxeles en la imagen de entrada y vértices de los modelos tridimensionales, podemos utilizar algún solucionador del problema *Perspective-n-Point*, con el cual se realiza la estimación de pose. Dado que las correspondencias contienen ruido, se utiliza RANSAC [7] para encontrar el modelo que minimice el error de reproyección a través de un proceso iterativo de ajuste y evaluación de hipótesis. El resultado es la pose codificada en una matriz de rotación y traslación.

Además de la primera estimación, DPOD tiene un sistema de refinamiento de pose que se entrena a partir de datos reales y sintéticos y recibe como entrada la imagen RGB y la pose estimada por el bloque de pose. A partir de la estimación de pose se genera una imagen artificial proyectando el modelo tridimensional con la pose estimada. Ambas imágenes entran a la red de refinamiento, cada una a un codificador que extrae un vector de características para cada imagen. Estos vectores son restados y el vector resultante entra a un nuevo codificador. A su vez, el vector resultante alimenta a tres decodificadores que se encargan cada uno de estimar un componente de la pose, tanto un vector de traslación como la matriz de rotación.

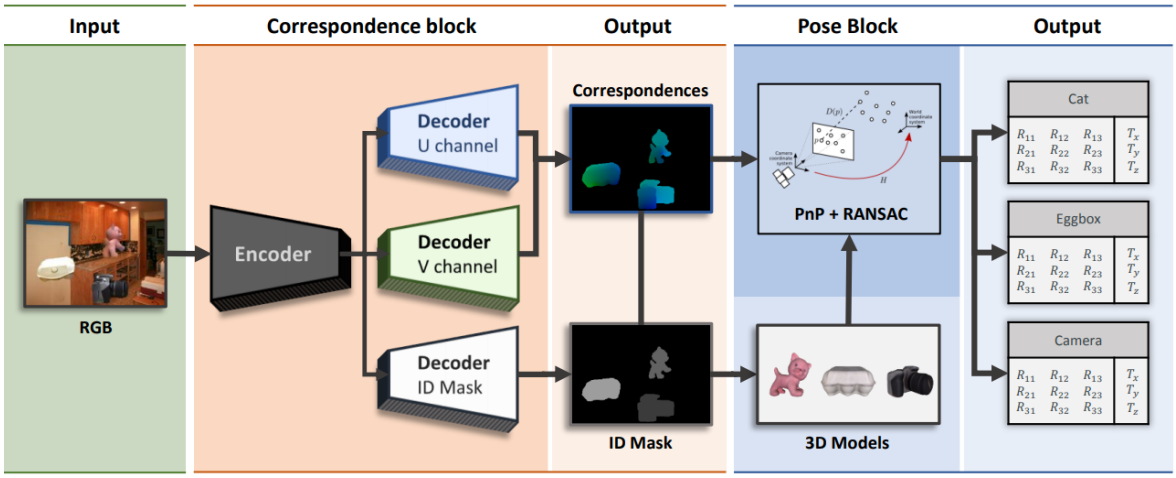

Figura 3.6: Arquitectura de DPOD. Figura tomada de [24].

A diferencia de DPOD, la arquitectura propuesta en esta tesis permite realizar aprendizaje de principio a fin. Si bien es cierto que el rendimiento de DPOD es superior al de muchas otras arquitecturas sobre LINEMOD, la red únicamente realiza la estimación de la segmentación semántica y de los mapas de correspondencias. El problema PnP es resuelto de manera convencional después a partir de los mapas obtenidos por la red. Esto implica que el proceso de estimación de pose mediante PnP no beneficia en absoluto el entrenamiento de la red mediante la actualización de los pesos y sesgos. La arquitectura propuesta permite que la información de la pose influya directamente en el entrenamiento de las capas convolucionales de la red, con lo cual se podría mejorar su desempeño, al utilizar el conocimiento codificado en el problema *Perspective-n-Point*.

3.5. *Differentiable Spatial to Numerical Transform*

Uno de los procesos dentro de la arquitectura de DPOD que no son diferenciables, es la obtención de los valores numéricos de las coordenadas del modelo tridimensional a partir de los canales de color. Este proceso está basado en operaciones de indizado no diferenciables. DSNT presenta una propuesta para poder realizar este proceso de forma totalmente diferenciable y que puede integrarse en redes para aprendizaje de principio a fin.

Para inferir las coordenadas numéricas de puntos de interés con redes neuronales convolucionales es usual que se utilicen mapas de calor o capas totalmente conexas. Sin embargo, ninguna de estas aproximaciones realmente permite una generalización espacial. DSNT ataca este problema sin agregar parámetros entrenables y es totalmente diferenciable, por lo que puede integrarse de forma natural con redes neuronales. La entrada de DSNT es un mapa de calor normalizado de un solo canal, representado por una matriz de $m \times n$. La normalización del mapa de calor consiste en que todos los elementos sean positivos y que sumen uno. La Figura 3.7 muestra el proceso de

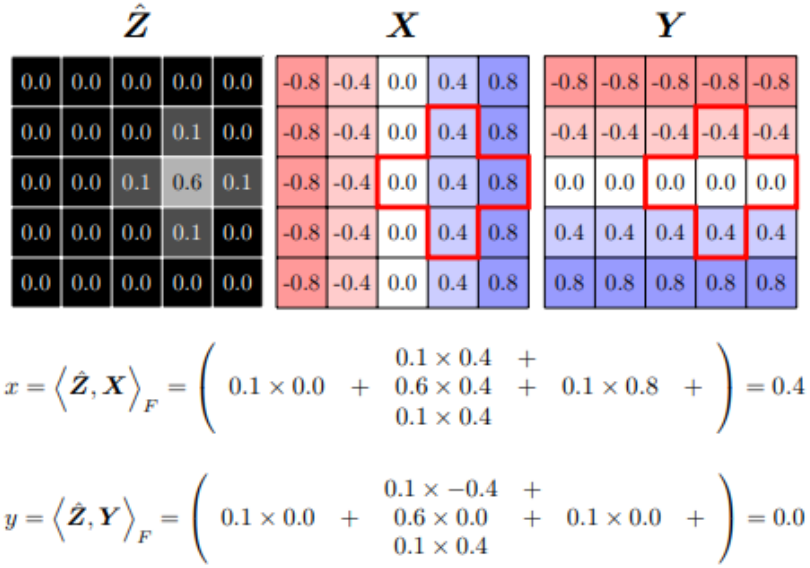

Figura 3.7: Estimación de coordenadas utilizando DSNT. Figura tomada de [15].

estimación de coordenadas.

Al integrar DSNT como un bloque dentro de la arquitectura propuesta, es posible aprovechar la estructura de DPOD e integrarlo naturalmente a la red para alimentar el solucionador de BPnP con las coordenadas resultantes.

3.6. *Refiners*

En conjunto con algunos de los sistemas para estimación de pose como DPOD [24], existen redes neuronales especializadas en mejorar la estimación de pose, conocidas como *refiners*. Para el caso de DPOD, existe un *refiner* (Figura 3.8) que obtiene como entrada la primera estimación de pose y se encarga de hacer un refinamiento adicional de la estimación. La entrada de esta red consiste en la pose estimada y el modelo tridimensional del objeto. Además de esto, se requieren dos imágenes de entrada. La primera es la misma que entra a DPOD, mientras que la segunda se obtiene al proyectar el modelo tridimensional del objeto con la pose estimada.

De esta forma, al primer codificador del *refiner* entra la imagen original y al segundo entra la imagen proyectada. Cada codificador extrae información de la imagen y posteriormente realiza una resta de los vectores. El vector resultante de esta resta entra a un tercer codificador que mapea las diferencias a un vector f, el cual se concatena con las coordenadas x y y de la estimación de pose de DPOD y entra a otro codificador que obtiene la estimación final de x y y. De forma paralela el vector f entra a otros dos codificadores, uno que obtiene la estimación final de z y otro que codifica la rotación. En conjunto estos tres últimos codificadores obtienen la estimación refinada de la pose.

Al utilizar este tipo de redes de refinamiento, se ha podido observar un incremento importante en el desempeño de redes como DPOD. Sin embargo, es importante considerar

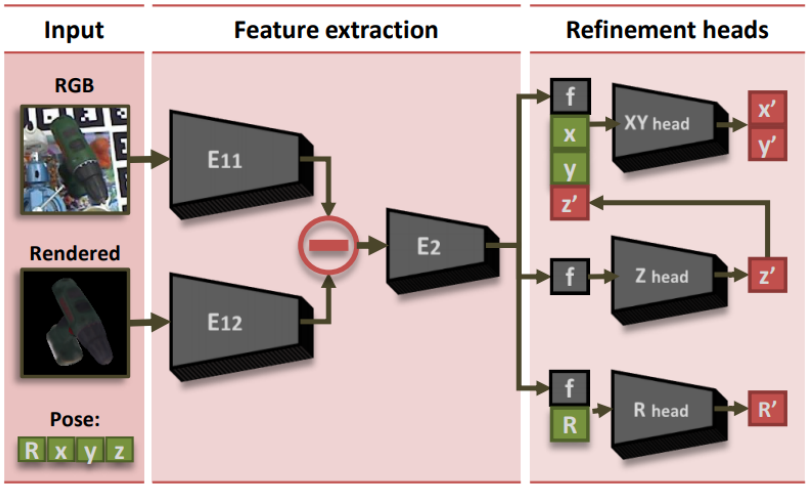

Figura 3.8: Red para el refinamiento de la pose. Figura tomada de [24].

que lo que hace esta red es introducir un sesgo, por lo cual debemos tener mucho cuidado en cómo se entrena y qué tan útil es realmente cuando se está realizando estimación de pose en ambientes con propiedades distintas a los ambientes de entrenamiento.

Capítulo 4

Metodología

En este capítulo se realiza y se fundamenta la propuesta específica de la tesis. Se explica cómo es que se llegó a esa arquitectura, el método de entrenamiento y los datos utilizados.

4.1. Bases de datos utilizadas

Como se comentó anteriormente, para poder realizar una comparación con los métodos del estado del arte, se utilizó la base de datos LINEMOD. Además de esto, también se generó una base de datos propia a partir de proyecciones artificiales de modelos tridimensionales.

4.1.1. LINEMOD

El conjunto de datos LINEMOD [10] consta de más de 1100 cuadros de video de 15 objetos sin textura para la evaluación de métodos de estimación de pose. En la Figura 4.1 se puede observar el modelo del objeto *duck*. Para cada video, se colocó el objeto de interés al centro de un tablero con marcas en los bordes. Estas marcas permitieron obtener la anotación de pose para cada uno de los cuadros de video. El proceso de captura se realizó de tal manera que los puntos de vista quedaran repartidos de manera uniforme sobre el hemisferio superior del objeto. En el cuadro 4.1 se enlistan los objetos que forman parte del conjunto de datos y el número de cuadros para cada objeto.

Además de los cuadros de video (Figura 4.2), el conjunto de datos contiene modelos tridimensionales de cada objeto, así como las anotaciones de pose para cada una de las imágenes. Los modelos tridimensionales contienen la información de miles de vértices pertenecientes a cada objeto, codificados mediante su posición con coordenadas tridimensionales y tres canales de color RGB.

Existen versiones ampliadas de este conjunto de datos, en las cuales se incluyen máscaras binarias para cada imagen, lo cual facilita el trabajo para redes que realizan segmentación semántica.

Objeto	Número de cuadros
Ape	1235
Bench Vise	1214
Driller	1187
Cam	1200
Can	1195
Iron	1151
Lamp	1226
Phone	1224
Cat	1178
Hole punch	1236
Duck	1253
Cup	1239
Bowl	1232
Box	1252
Glue	1219

Cuadro 4.1: Número de cuadros de video para cada objeto en LINEMOD.

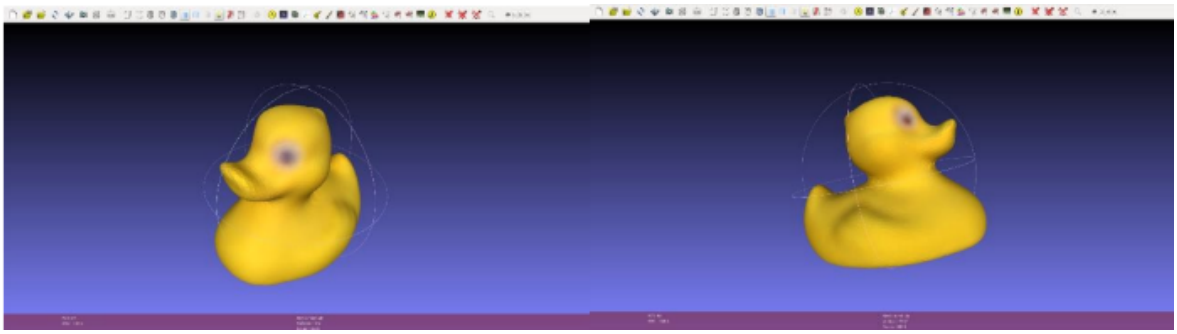

Figura 4.1: Modelo del objeto Duck del conjunto de datos LINEMOD

Figura 4.2: Imagen del objeto Cat tomada de LINEMOD.

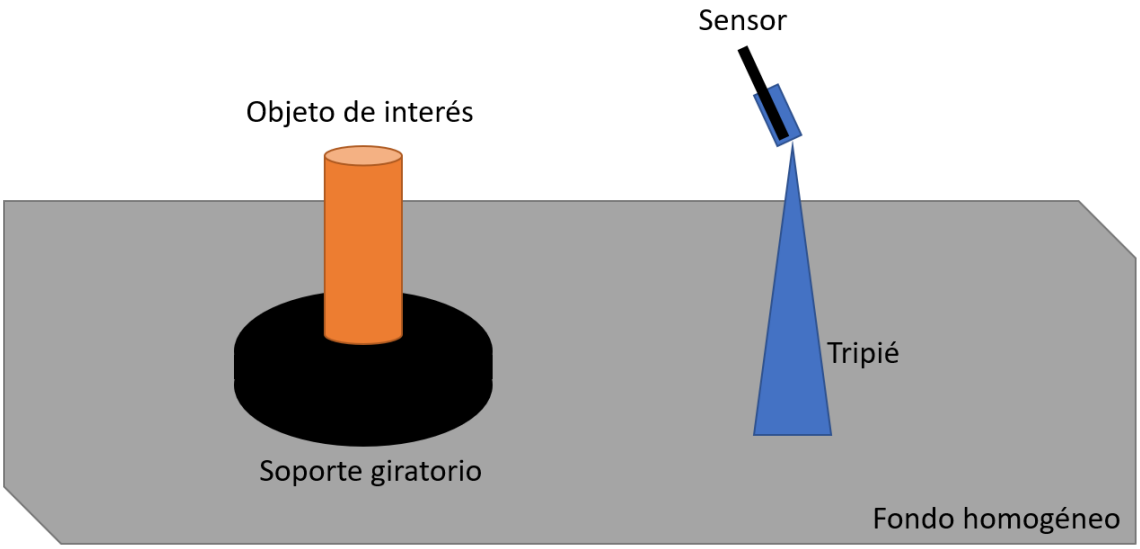

Figura 4.3: Configuración para la generación de modelos tridimensionales.

4.1.2. Generación de proyecciones artificiales a partir de modelos 3D

Además de utilizar la base de datos estándar de LINEMOD, se creó una base de datos propia. Esta base contiene poco más de 2000 imágenes de dos objetos. Para poder hacer estas imágenes, se realizaron proyecciones artificiales de objetos tridimensionales. A cada imagen resultante se le agregó como fondo una imagen perteneciente a MS-COCO [13].

Para los modelos tridimensionales de los objetos se utilizó un iPhone®12 Pro Max. Este teléfono inteligente cuenta con un sensor de profundidad que utiliza luz estructurada y nos permite generar modelos tridimensionales de objetos. En la Figura 4.3 se puede observar la configuración que se utilizó para la generación de modelos tridimensionales. Utilizando la aplicación 3D Scanner App™, podemos generar una nube de puntos de una escena al mover el celular alrededor del objeto de interés. Para facilitar el proceso se utilizó una base giratoria sobre la cual se colocó el objeto, de esta forma el teléfono inteligente se mantiene fijo en un tripié mientras el objeto rota.

Una vez obtenida la nube de puntos (Figura 4.4), tenemos las coordenadas xyz y los canales de color RGB. El escaneo inicial generado por la aplicación contiene información del fondo de la escena que no pertenece al objeto, por lo que es necesario eliminar el ruido (Figura 4.5). Además, para poder realizar las proyecciones es necesario generar las normales y hacer una reconstrucción de las caras utilizando un software de renderizado tridimensional como Meshlab™(Figura 4.6). Después de este proceso, los modelos tridimensionales de los objetos pueden ser utilizados para generar las proyecciones artificiales. Este proceso se realizó utilizando el repositorio *Nocs-Renderer*, desarrollado por los autores de DPOD [24], disponible en *https://github.com/zakharos/nocs-renderer*. A partir de un archivo PLY, Nocs-Renderer realiza un muestreo de la nube de puntos

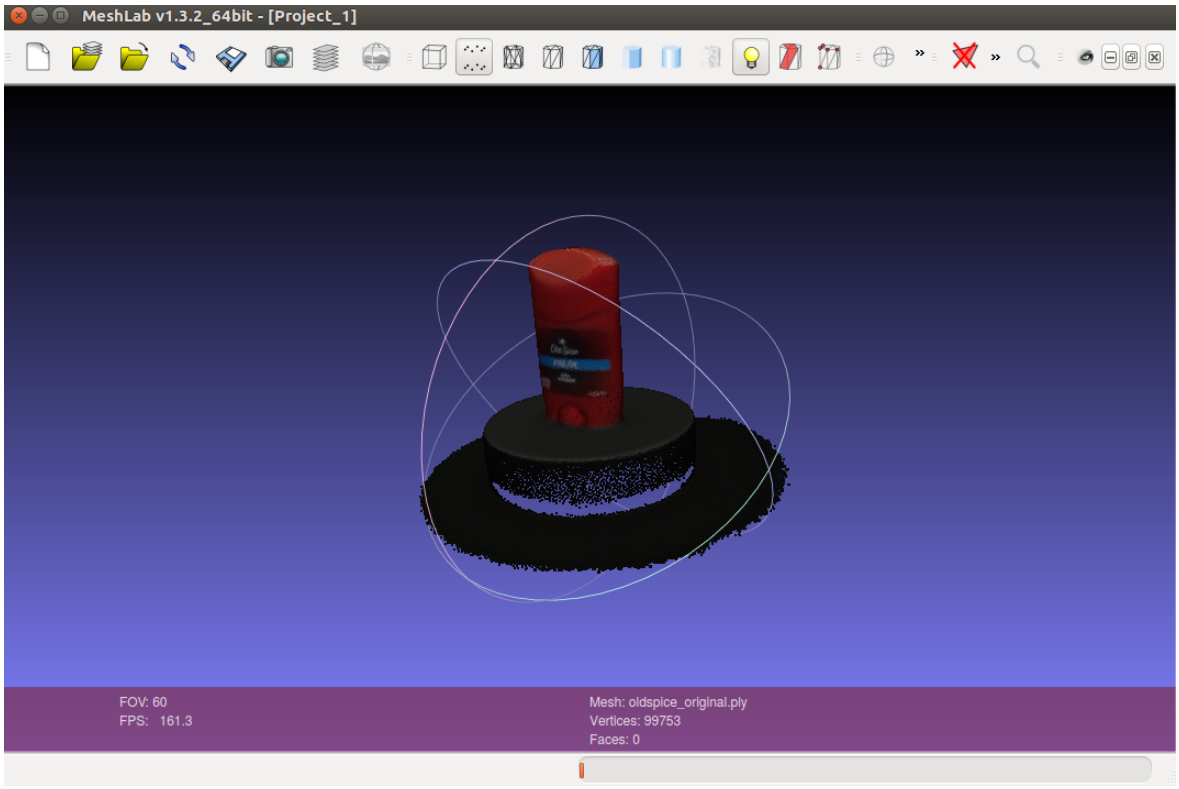

Figura 4.4: Nube de puntos exportada de 3D Scanner App sin procesar.

y proyecta el objeto dada una pose y los parámetros intrínsecos de la cámara (se utilizaron los mismos parámetros proporcionados en LINEMOD [10]). Para el caso de entrenamiento con DPOD [24], es posible también generar imágenes que contengan la información de los canales UV o UVW. Para obtener los mapas, se puede realizar una proyección cilíndrica o esférica del modelo sobre la textura y así asignar a cada vértice un color único. Para el caso de los mapas UVW este proceso se puede hacer de una manera más sencilla, ya que sabemos que la combinación de coordenadas de los vértices es única y entonces podemos normalizar las coordenadas de cada vértice para así obtener tres canales de color.

4.2. Desarrollo de las arquitecturas

Para poder presentar las arquitecturas propuestas, es fundamental comprender los subproblemas involucrados, específicamente la detección de características, la asignación de correspondencias y la estimación de la pose.

Como se mencionó en capítulos anteriores, existen diversos métodos para la extracción de características. Algunos de los métodos basados en características locales más conocidos incluyen SIFT [14], SURF [2] y ORB [18], pero también existen aproximaciones basadas en aprendizaje profundo como SuperPoint [6].

Después de obtener las características por alguno de los métodos antes mencionados,

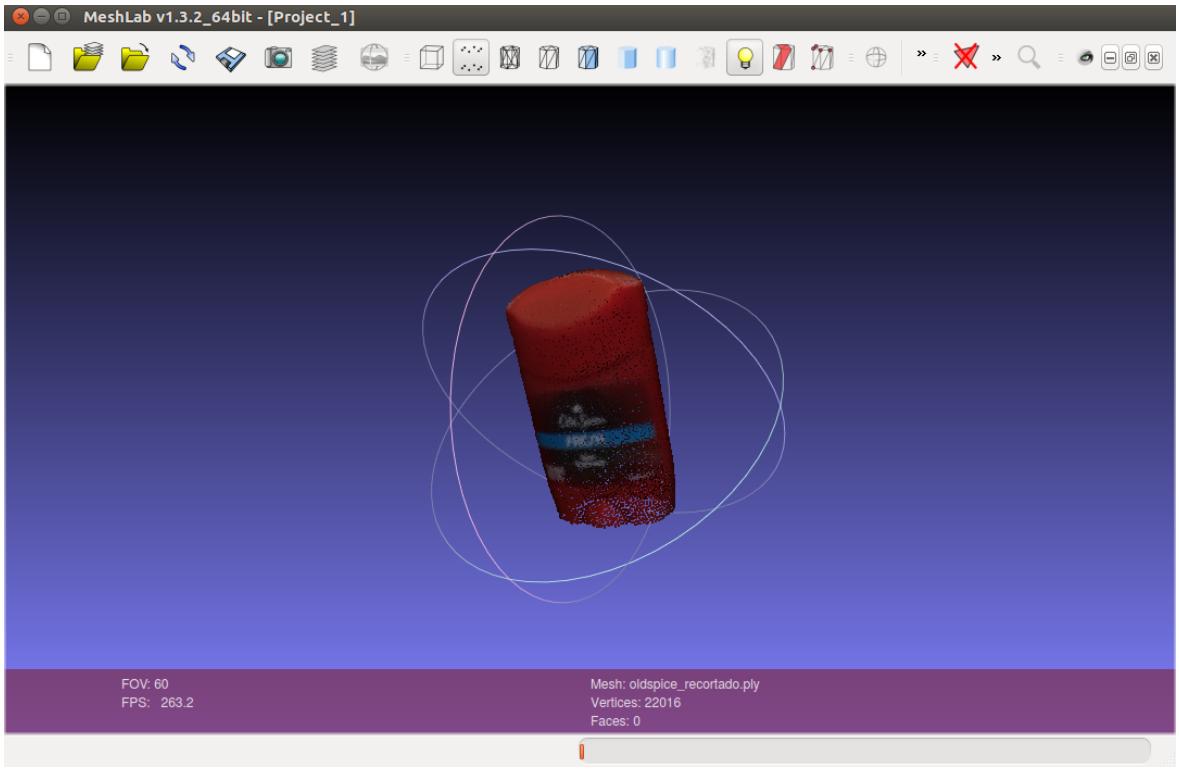

Figura 4.5: Nube de puntos recortada manualmente mediante Meshlab.

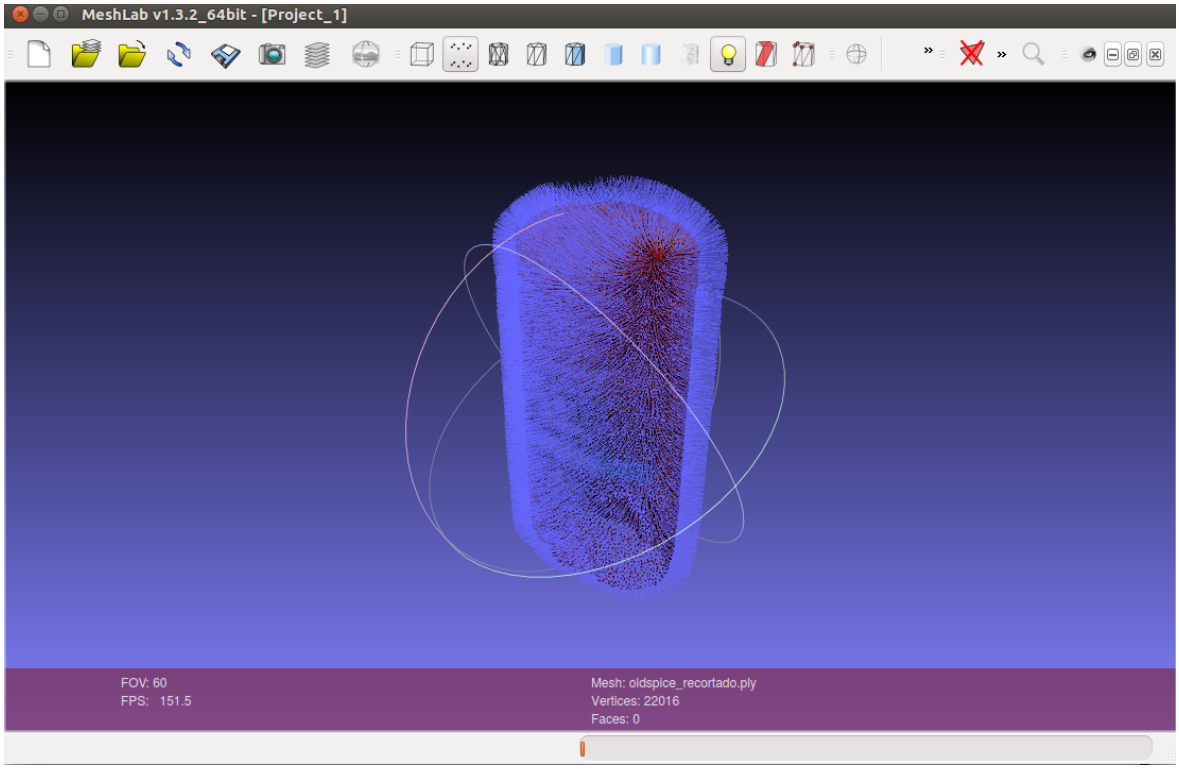

Figura 4.6: Normales de la nube de puntos calculadas con Meshlab.

se requiere encontrar las correspondencias. Esta tarea usualmente se lleva a cabo a través de algoritmos de búsqueda de vecino más cercano, como fuerza bruta o Randomized KD-Trees, que utilizan una distancia entre los descriptores para encontrar las correspondencias. Sin embargo, existen también métodos basados en aprendizaje profundo como SuperGlue [19], que codifica las correspondencias mediante grafos con pesos variables.

Finalmente, con las correspondencias solucionamos el problema PnP siguiendo algunos de los métodos presentados como EPnP+RANSAC y así obtenemos la pose del objeto de interés.

Una vez presentados los subproblemas procedamos a analizar el comportamiento de los sistemas de estimación de pose de principio a fin basados en aprendizaje profundo. El uso de redes neuronales convolucionales ha permitido mejorar el desempeño en la tarea de estimación de pose, principalmente en la extracción de características. A diferencia de métodos como SIFT [14], las redes neuronales convolucionales permiten extraer características de objetos sin textura. En la búsqueda de correspondencias, las redes neuronales profundas también han mejorado los rendimientos de los algoritmos mencionados anteriormente.

Sistemas como SuperGlue [19], SuperPoint [6] y DPOD [24] únicamente resuelven parcialmente el problema de estimación de pose con redes neuronales profundas. SuperPoint se encarga de la extracción de puntos característicos y SuperGlue se encarga de encontrar correspondencias entre conjuntos de características. Por su parte, DPOD [24] resuelve mediante aprendizaje profundo el problema de detección de correspondencias, mientras que el proceso de estimación de pose se hace de manera tradicional utilizando el solucionador de PnP con RANSAC.

La principal razón por la que no se realiza el aprendizaje de la estimación de pose de principio a fin es porque PnP y RANSAC no son diferenciables. Para solucionar este problema, se introdujo BPnP [4], una versión del solucionador PnP que permite propagar los gradientes. Una de las propuestas de esta tesis es integrar BPnP a una arquitectura basada en DPOD para hacer aprendizaje de principio a fin. Esto permite aprovechar el conocimiento geométrico que se tiene del problema *Perspective-n-Point* en conjunto con las características que pueden extraer las redes neuronales convolucionales.

En los últimos años se han propuesto diversas arquitecturas que aprenden la estimación de la pose de principio a fin pero no resuelven de forma explícita el problema PnP, tal es el caso de PoseCNN [23], ConvPoseCNN [3], YOLO6D [22] y PVNet [16]. Todas estas arquitecturas, a pesar de que tienen un rendimiento que ha llegado a ser parte del estado del arte en los últimos años, no aprovechan el conocimiento geométrico que se tiene del problema PnP. Al realizar regresión directamente de las características a la pose, de forma subyacente estas arquitecturas mapean las características a la pose [4]. Esto significa que la red no puede generalizar correctamente y es propenso a realizar sobreajuste [20].

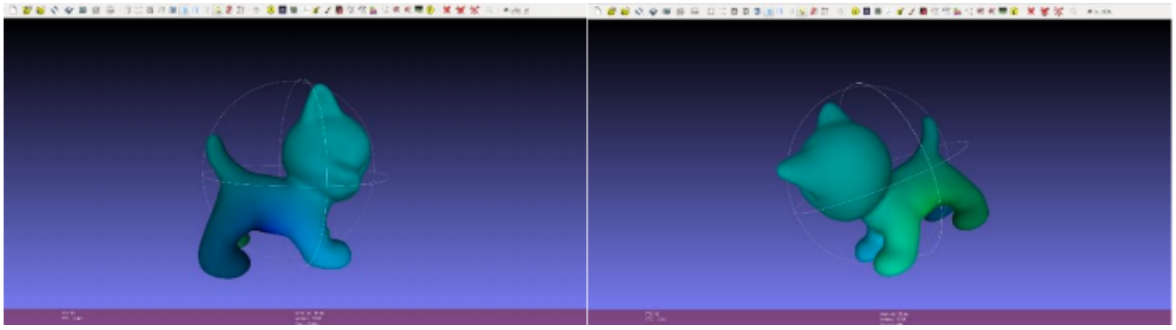

Figura 4.7: Modelo con una textura UV de un objeto del conjunto de datos LINEMOD

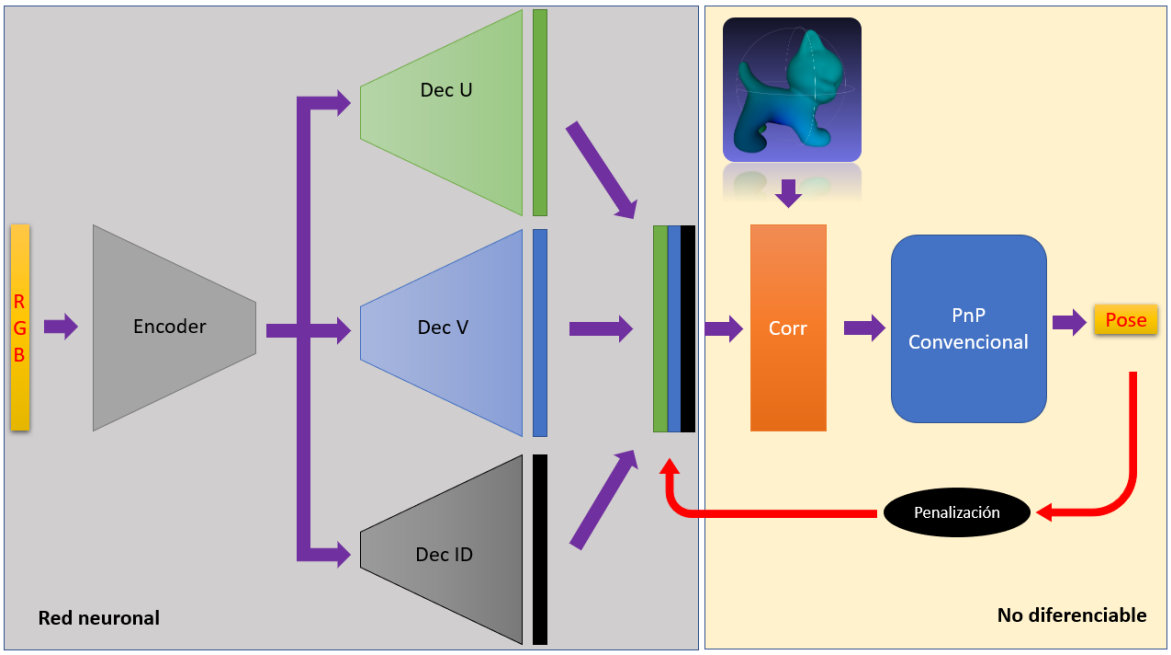

Figura 4.8: DPOD con penalización por pose.

4.2.1. DPOD con penalización

La primer arquitectura propuesta consiste en una penalización en la función de pérdida de DPOD.

Debido a que la función de pérdida de DPOD está basada puramente en la calidad de las correspondencias UV (Figura 4.7) obtenidas del decodificador, la propuesta de DPOD con penalización por pose agrega un término a la función de pérdida como se observa en la Figura 4.8.

Con esta modificación a la función de pérdida de DPOD, se toma en cuenta la diferencia entre los puntos proyectados en el modelo tridimensional del objeto dadas la pose anotada y la pose predicha.

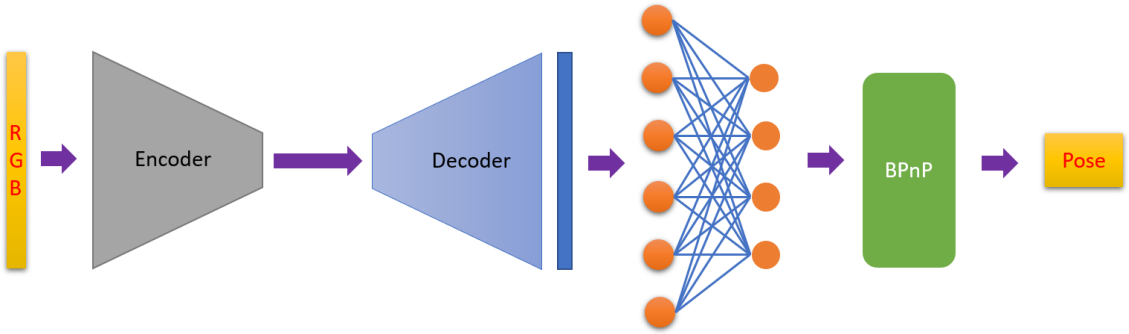

Figura 4.9: Segunda arquitectura propuesta, FC-BPnP.

4.2.2. FC-BPnP

A pesar de que DPOD con penalización integra un término que permite tomar en cuenta la pose para el ajuste de los parámetros, el aprendizaje no es de principio a fin. Específicamente, el mapeo entre las combinaciones de colores UV y los puntos tridimensionales se realiza a partir de un proceso rígido de indizado que no es diferenciable. Además, se siguen utilizando RANSAC y PnP, los cuales involucran operaciones que tampoco son diferenciables.

En FC-BPnP (Figura 4.9) se agrega un codificador-decodificador que se encarga de extraer las características de la imagen, y mediante una serie de capas totalmente conexas, se obtienen las coordenadas de cada uno de los puntos de interés en la imagen. A partir de esto, se integra BPnP, lo cual permite obtener la pose en un proceso totalmente diferenciable.

Dado que las capas totalmente conexas cuentan con muchos pesos entrenables al aumentar el número de neuronas, fue necesario submuestrear la nube de puntos. Como cada modelo tridimensional de LINEMOD contiene miles de vértices, se submuestreó a unas cuantas decenas para obtener un compromiso entre la calidad y la complejidad. A En esta tesis de forma arbitraria se tomaron 27 puntos representativos de cada nube utilizando Meshlab™. La red se encarga de darnos las coordenadas de estos 27 puntos dentro de la imagen. Una vez que tenemos la posición bidimensional de los puntos, se mapean a las coordenadas de los puntos tridimensionales del modelo submuestreado y se ingresan las coordenadas de ambas nubes de puntos a BPnP, de esta forma propagamos el error desde la pose y el entrenamiento se hace de principio a fin. El bloque BPnP tiene dos componentes. Al momento de hacer un *forward pass*, utiliza las funciones PnP+RANSAC, sin embargo, al hacer retropropagación, se utiliza una función personalizada de Autograd de PyTorch, que permite calcular los gradientes y propagarlos al resto de la red.

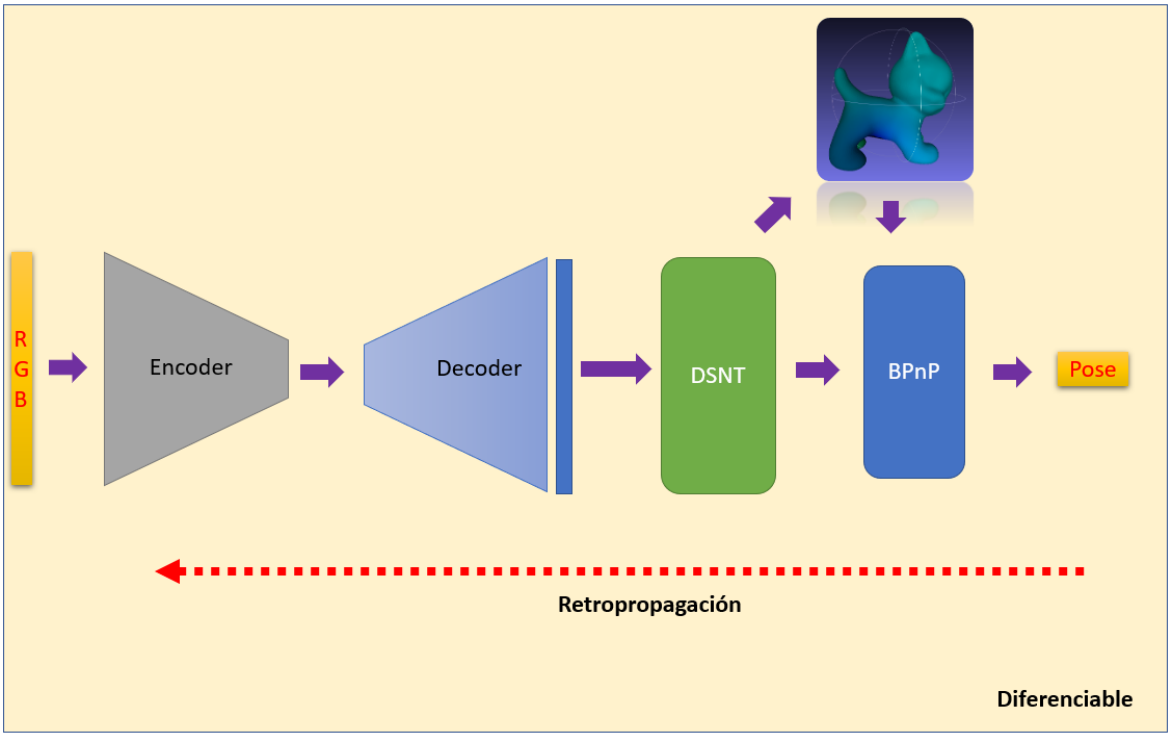

Figura 4.10: Tercera arquitectura propuesta, DSNT-BPnP.

4.2.3. DSNT-BPnP

Las capas totalmente conexas en FC-BPnP, si bien pueden realizar la tarea, no son la mejor opción para poder realizar manejo de coordenadas dentro de una red neuronal ya que carecen de generalización espacial inherente [15]. Por lo tanto se propone introducir DSNT [15] para sustituir las capas totalmente conexas como se muestra en la Figura 4.10.

DSNT permite mapear de forma diferenciable mapas de calor a un conjunto de coordenadas mediante la aplicación de un promedio ponderado con pesos específicos para esta tarea. Al ingresar esta información al módulo BPnP, se obtiene la pose del objeto. Con esta nueva propuesta se puede realizar aprendizaje de principio a fin, además de que se reduce significativamente la cantidad de información requerida de cada objeto durante el entrenamiento, a diferencia de DPOD, que requiere los mapas UV. El hecho de no necesitar generar texturas UV para cada modelo tridimensional además de las anotaciones de pose por cada imagen supone un ahorro de tiempo considerable.

El entrenamiento de las redes propuestas se puede realizar tanto con imágenes originales como con imágenes sintéticas. Las imágenes originales utilizan directamente imágenes de la base de datos LINEMOD con las anotaciones correspondientes sobre la pose. Sin embargo, dado que contamos con los modelos tridimensionales de cada objeto de LINEMOD, podemos realizar proyecciones artificiales como se explicó anteriormente.

Las proyecciones de los objetos generan imágenes RGB y sus correspondientes normales, mapas UV o UVW y mapas de profundidad. A partir de las imágenes

generadas por las proyecciones y utilizando los fondos de MS-COCO, obtenemos un conjunto de datos con el cual podemos entrenar cualquiera de las redes propuestas.

Capítulo 5

Resultados

En este capítulo se presentan los resultados experimentales, así como las métricas que se utilizaron para evaluar el desempeño del modelo de estimación de pose y su comparación con algunos de los sistemas de estimación de pose del estado del arte.

5.1. Métricas de evaluación

La evaluación de los sistemas de estimación de pose se hace mediante una métrica llamada distancia promedio (ADD) [23], mediante la medición de la diferencia de ángulos y traslaciones o mediante la distancia de Hausdorff. En esta tesis nos enfocamos en la métrica ADD para facilitar la comparación con los métodos de estimación de pose mencionados en el capítulo referente al estado del arte. ADD es el promedio de la distancia euclidiana entre los vértices del modelo transformados con la pose predicha y la pose anotada.

Dadas la rotación y traslación reales ($\boldsymbol{R}$ y $\boldsymbol{t}$) y las estimadas ($\hat{\boldsymbol{R}}$ y $\hat{\boldsymbol{t}}$). La distancia promedio calcula el promedio de la distancia entre los puntos tridimensionales del modelo transformados de acuerdo a la pose real y a la pose estimada

$$ADD = \frac{1}{m} \sum_{x \in \boldsymbol{M}} ||(\boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}) - (\hat{\boldsymbol{R}}x + \hat{\boldsymbol{t}})|| \tag{5.1}$$

donde $\boldsymbol{M}$ es el conjunto de puntos tridimensionales del modelo y m es el número de puntos. La pose se considera correcta si la distancia promedio es menor que algún umbral definido. Usualmente el umbral es el 10 % del diámetro del modelo. Para objetos simétricos, se utiliza la métrica ADD-S, que evita la ambiguedad entre las correspondencias

$$ADD - S = \frac{1}{m} \sum_{x_1 \in \boldsymbol{M}} \min_{x_2 \in \boldsymbol{M}} ||(\boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}) - (\hat{\boldsymbol{R}}\boldsymbol{x} + \hat{\boldsymbol{t}})|| \tag{5.2}$$

Object	YOLO6D	PVNet	DPOD	DSNTBPNP
Ape	21.62	43.62	53.28	56.36
Benchvise	81.80	99.90	95.34	70.91
Cam	36.57	86.86	90.36	84.54
Can	68.80	95.47	94.10	96.36
Cat	41.82	79.34	60.38	91.58
Driller	63.51	96.43	97.72	69.09
Duck	27.23	52.58	66.01	82.72
Eggbox	69.58	99.15	99.72	91.81
Glue	80.02	95.66	93.83	89.09
Holepuncher	42.63	81.92	65.83	97.27
Iron	74.97	98.88	99.80	92.72
Lamp	71.11	99.33	88.11	81.17
Phone	47.74	92.41	74.24	83.36
Promedio	55.95	86.27	82.98	83.61

Cuadro 5.1: Cuadro comparativo del modelo desarrollado contra otros modelos que no utilizan refinamiento utilizando datos reales. El cuadro muestra el porcentaje de estimaciones correctas con una tolerancia del 10 % del diámetro del objeto.

5.2. Entrenamiento con datos reales, LINEMOD

Para poder realizar una comparación justa con otros métodos, se utilizó la métrica ADD con la misma tolerancia con la que fue utilizada en los artículos correspondientes. En el cuadro 5.1 se muestra la comparación entre el rendimiento del modelo desarrollado y el de algunos de los métodos que pertenecen al estado del arte como YOLO6D, PVNet y DPOD. La evaluación está basada en la métrica ADD-10 que corresponde a un umbral del 10 % del diámetro del objeto. Los datos utilizados para la evaluación es el conjunto de datos LINEMOD. Se puede observar que la evaluación se hace de forma separada para cada objeto en la mayoría de los casos.

Existe una diferenciación entre los métodos que ocupan algún tipo de refinamiento y aquellos que no lo tienen. Esta tesis se enfocó en el análisis de los objetos pertenecientes a LINEMOD, sin embargo es necesario mencionar que no se está utilizando una red de refinamiento, dado que cae fuera de los límites de este trabajo. La ventaja de los modelos sin red de refinamiento es que evitan aprender características contextuales del conjunto de entrenamiento.

Como se puede observar en la tabla, el modelo tiene un desempeño superior a muchas de las arquitecturas que pertenecen al estado del arte, principalmente en los objetos *cat* y *duck*. De las arquitecturas que no tienen refinamiento, la única que la supera es PVNet. Los resultados presentados en el Cuadro 5.1 realizan la comparación con redes que tampoco tienen refinamiento. Es realista considerar que al implementar un proceso de refinamiento adicional los resultados obtenidos sean comparables o incluso superiores tanto a DPOD con refinamiento como a PVNet.

Figura 5.1: Resultados cualitativos para los primeros seis modelos entrenados con datos reales de LINEMOD. La estimación de pose real y la estimada se muestran en azul y verde, respectivamente.

Figura 5.2: Resultados cualitativos para los segundos seis modelos entrenados con datos reales de LINEMOD. La estimación de pose real y la estimada se muestran en azul y verde, respectivamente.

Object	SSD6D	AAE	DPOD	DSNTBPNP
Ape	2.6	3.96	37.22	45.45
Benchvise	15.1	20.92	66.76	60.73
Cam	6.1	30.47	24.22	51.81
Can	27.3	35.87	52.57	54.74
Cat	9.4	17.90	32.36	36.36
Driller	12.0	23.99	66.60	25.12
Duck	1.3	4.86	26.12	42.72
Eggbox	2.8	81.01	73.35	87.27
Glue	3.4	45.49	74.96	61.91
Holepuncher	3.1	17.60	24.50	45.45
Iron	14.6	32.03	85.02	86.36
Lamp	11.4	60.47	57.26	47.30
Phone	9.7	33.79	29.08	62.72
Promedio	9.1	28.65	50.0	54.46

Cuadro 5.2: Cuadro comparativo del modelo desarrollado contra otros modelos que no utilizan refinamiento utilizando datos sintéticos. El cuadro muestra el porcentaje de estimaciones correctas con una tolerancia del 10 % del diámetro del objeto.

5.3. Entrenamiento con datos sintéticos, LINEMOD

En las figuras 5.3 y 5.4 se muestran imágenes correspondientes a la estimación de pose para los objetos pertenecientes al conjunto de datos LINEMOD. Las imágenes muestran las poses predichas en verde y las reales en azul. En las imágenes se puede observar como la diferencia entre ambas poses es muy poca incluso utilizando datos sintéticos para entrenar.

Para tareas como manipulación robótica, una tolerancia como la de ADD-10 es más que suficiente para poder realizar la manipulación de un objeto que una persona podría manipular con una mano. Ciertamente a pesar de que la métrica ADD-10 es la mejor referencia para medir el rendimiento de un sistema de estimación de pose, es posible evaluar los sistemas con métricas menos estrictas como ADD-30 siempre y cuando la tarea en la que será aplicada permita esa tolerancia en la estimación.

En la práctica, al entrenar objetos nuevos es mucho más sencillo entrenar con datos sintéticos que con datos reales, dado que realizar las anotaciones de pose para un objeto no conocido supone un gran esfuerzo. Al realizar la proyección de forma artificial, únicamente necesitamos generar un modelo tridimensional del objeto de interés, que puede realizarse fácilmente con ayuda de un sensor especializado o de un teléfono inteligente con las especificaciones adecuadas.

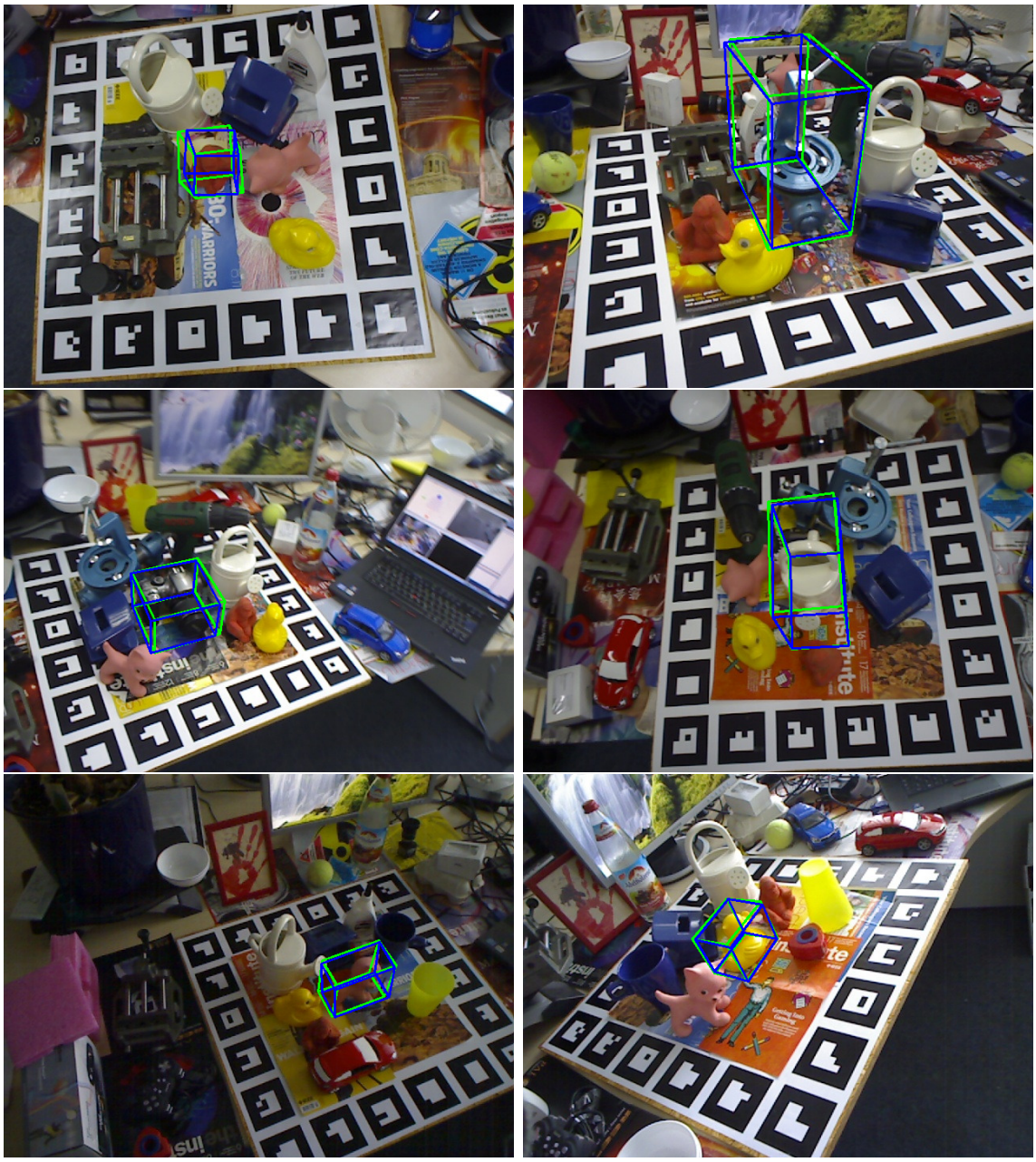

Figura 5.3: Resultados cualitativos para los modelos entrenados con datos sintéticos de LINEMOD. La estimación de pose real y la estimada se muestran en azul y verde, respectivamente.

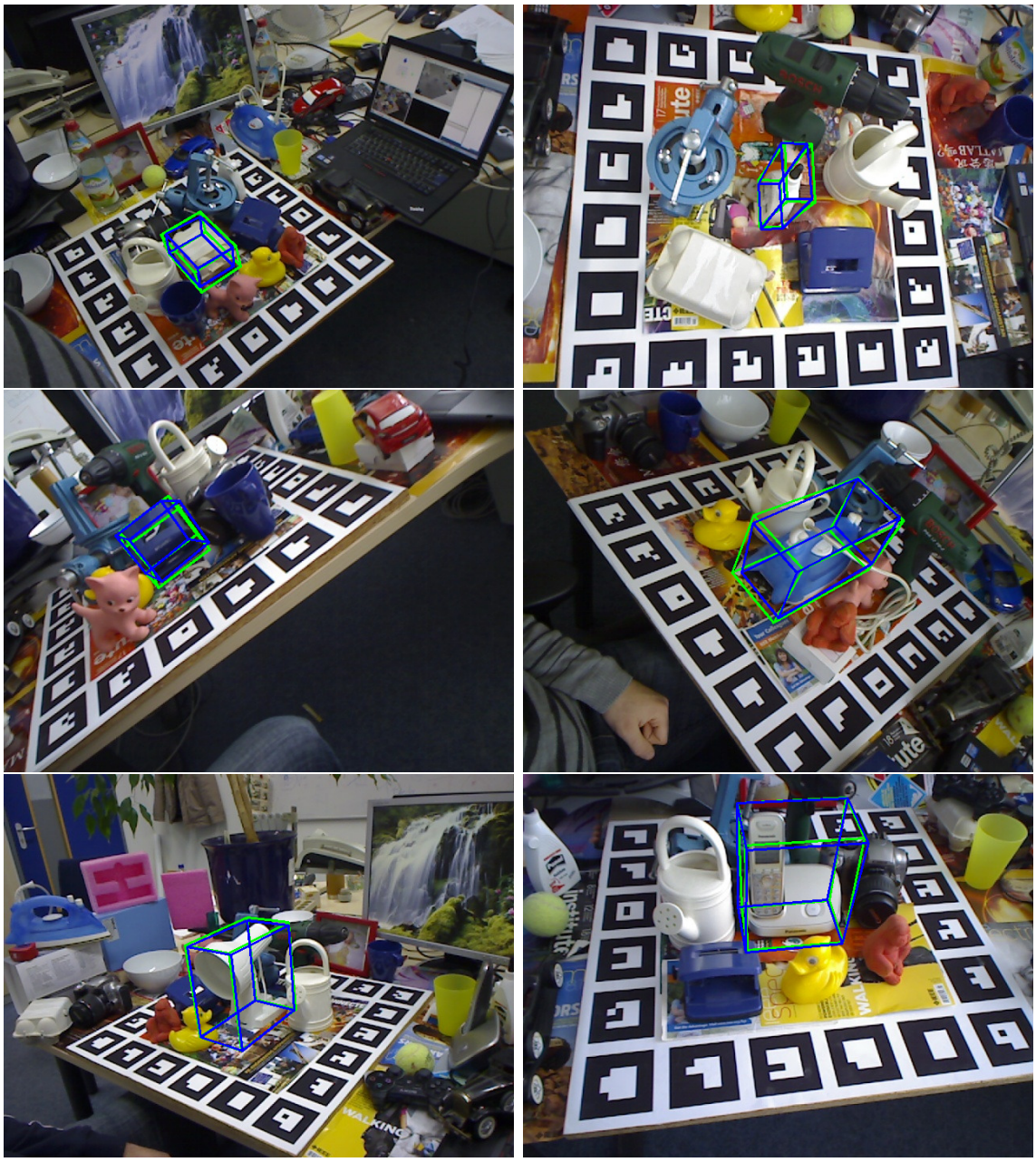

Figura 5.4: Resultados cualitativos para los modelos entrenados con datos sintéticos de LINEMOD. La estimación de pose real y la estimada se muestran en azul y verde, respectivamente.

ADD	ADD-10	ADD-30	ADD-50
Oldspice™	36.14	40.1	60.91
Lector	46.29	84.25	88.9
Caja	58.18	78.2	83.63

Cuadro 5.3: Cuadro comparativo de los modelos desarrollados utilizando objetos propios usando el método propuesto. El cuadro muestra el porcentaje de estimaciones correctas con una tolerancia del 10 % del diámetro del objeto.

5.4. Entrenamiento con objetos propios

El entrenamiento con objetos propios es de especial importancia para la comunidad de estimación de pose en robótica de servicio, dado que una de las tareas del sistema de visión es poder estimar la posición relativa de los objetos conocidos respecto del robot. Para poder lograr este fin es posible entrenar el sistema mediante la generación de un modelo tridimensional y una serie de proyecciones artificiales.

Para la generación del modelo se utilizó un iPhone 12®Pro Max, que cuenta con un sensor de profundidad y es capaz de generar modelos tridimensionales a color con una tolerancia milimétrica. El proceso se puede realizar en un par de minutos con ayuda de un tripié y un fondo homogéneo.

Dado que el modelo generado contiene información de fondo que no nos es útil, es necesario limpiar los vértices del modelo que no pertenecen al objeto de interés y posteriormente calcular las normales de la nube de puntos con ayuda de un programa de modelado tridimensional como Meshlab™. La nube de puntos debe estar centrada en el origen del sistema de coordenadas para poder realizar la proyección adecuadamente.

Posteriormente, se realiza la proyección del modelo con una serie de poses de preferencia repartidas homogéneamente alrededor del objeto en el hemisferio superior, donde es más probable que sea observado. Una vez generadas las imágenes, podemos generar nuestro conjunto de datos de entrenamiento y podemos entrenar el sistema de estimación de pose.

Además de tener la capacidad de estimar la pose de objetos propios, es muy valorado dentro de la comunidad de robótica de servicio tener la capacidad de generarlos de una manera sistemática para que pueda ser útil en escenarios como certámenes de robótica donde el proceso de adquisición de datos, procesamiento de las imágenes y entrenamiento de la red se deben hacer en tan solo unas horas.

En el cuadro 5.3 se presentan los resultados obtenidos para tres objetos propios utlizados: una barra de Oldspice™, un lector de radiofrecuencia y una caja de cartón. Estos resultados se obtuvieron siguiendo la metodología mencionada anteriormente. Podemos ver que a pesar de que el entrenamiento se realizó únicamente con datos sintéticos, el rendimiento con ADD-10 es comparable con los resultados obtenidos en LINEMOD, por lo cual este sistema podría ser utilizado para el entrenamiento de objetos nuevos en un sistema de visión para un robot móvil autónomo.

Figura 5.5: Resultados cualitativos para el modelo propio del objeto Oldspice™. La estimación de pose real y la estimada se muestran en azul y verde, respectivamente.

Figura 5.6: Resultados cualitativos para el modelo propio del objeto Caja. La estimación de pose real y la estimada se muestran en azul y verde, respectivamente.

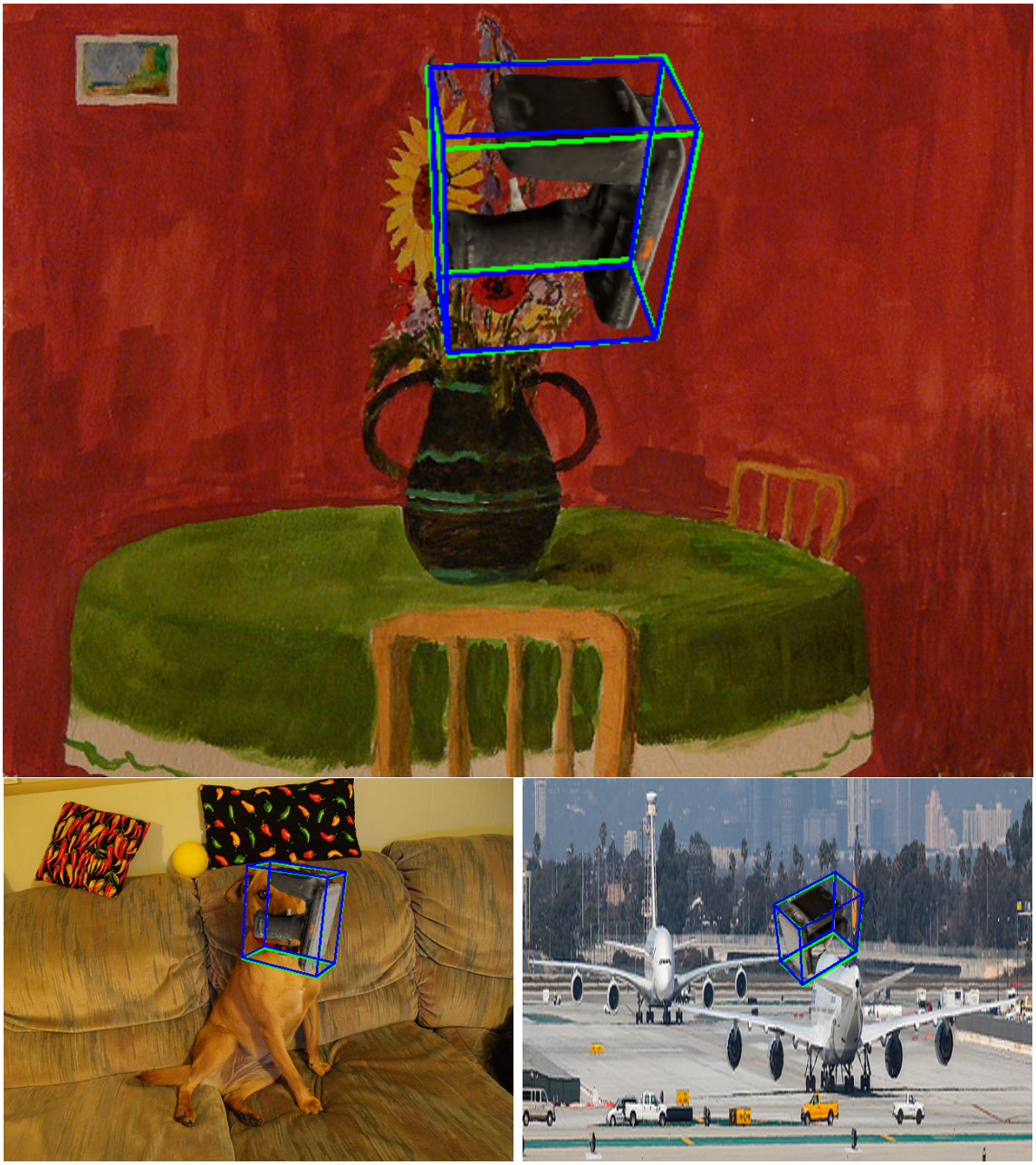

Figura 5.7: Resultados cualitativos para el modelo propio del objeto Lector. La estimación de pose real y la estimada se muestran en azul y verde, respectivamente.

Figura 5.8: Detecciones erróneas.

5.5. Discusión

El sistema de estimación de pose fue capaz de aprender de principio a fin la estimación de pose 6D codificando el conocimiento de una aproximación PnP. Además, el sistema propuesto superó el rendimiento de DPOD, PVNET y YOLO6D en los objetos *Ape*, *Can*, *Cat*, *Duck* y *Holepuncher*, además de que el rendimiento obtenido al promediar todos los objetos es superior.

A pesar de que el desempeño del modelo fue bueno, es importante resaltar algunos de los escenarios en los que le fue difícil realizar la estimación de pose y que fue superado por algunos otros métodos. Cuando el objeto tiene una geometría compleja, el submuestro del modelo provoca que sea más complicado para el sistema obtener la posición de los vértices. Es por ello que es recomendable utilizar un número superior de vértices para objetos complejos a pesar de que el tiempo de procesamiento pueda aumentar para poder mejorar el desempeño del sistema de estimación de pose. En la figura 5.8 se puede observar este escenario.

Cabe destacar también que a diferencia de métodos como DPOD, para poder realizar el entrenamiento únicamente es necesario el modelo tridimensional del objeto y una serie de poses propuestas para el entrenamiento que se puede generar mediante el sistema de proyección. En el caso de DPOD, es necesario realizar un mapa con textura UV para cada objeto, además de la pose de cada objeto. Esto supone una contribución dado que el esfuerzo para entrenar es menor y tenemos un rendimiento comparable o superior a DPOD.

Capítulo 6

Conclusiones

En esta tesis se presentó una nueva arquitectura de red neuronal convolucional para realizar la estimación de pose de objetos a partir únicamente de una imagen RGB como entrada. Se describieron los métodos pertenecientes al estado del arte actual, así como sus ventajas y desventajas con respecto al modelo propuesto. Se presentaron los distintos métodos para la resolución del problema *Perspective-n-Point*, se explicó la naturaleza no diferenciable del problema al ser utilizado en conjunto con RANSAC.

Se presentó la propuesta de arquitectura de la red neuronal convolucional, que permite el aprendizaje de estimación de pose de principio a fin al utilizar BPnP, una versión de PnP que permite la propagación de gradientes desde la pose hacia las correspondencias, lo cual permite aprovechar el conocimiento geométrico que se tiene acerca del problema de estimación de pose. Esto aumenta la capacidad de generalización al evitar hacer únicamente regresión de las características a la pose o a alguno de sus componentes. A pesar de mostrar una mejora en el rendimiento sobre el conjunto de datos de LINEMOD, es importante resaltar que el tiempo de procesamiento aumentó, así como la complejidad en tiempo y memoria.

La red fue capaz de aprender de principio a fin la estimación de pose 6D codificando el conocimiento de una aproximación PnP. Además de que el desempeño de la red propuesta bajo la métrica ADD-10 demuestra que se tiene un rendimiento comparable o superior para la mayoría de los objetos pertenecientes al conjunto de datos LINEMOD.

Se puede observar también que la calidad de las estimaciones permitiría a un sistema robótico manipular objetos que podemos encontrar en nuestro entorno cotidiano en la casa o en la oficina, como lo suponen las tareas de RoboCup@home™. Para poder determinar si el sistema es útil para ambientes distintos, sería recomendable realizar el entrenamiento con objetos con diferentes tamaños y características geométricas.

La incorporación de un sistema de refinamiento a la red permitiría en principio mejorar el desempeño de la red bajo conjuntos de datos como LINEMOD, sin embargo, esto queda fuera del alcance de esta tesis.

Figura 6.1: Robot Golem III realizando un proceso de estimación de pose que posteriormente le permitirá hacer manipulación robótica.

6.1. Aplicaciones y trabajo a futuro

Como se mencionó en el primer capítulo, la estimación de pose es un problema fundamental en visión computacional. En el campo de la robótica la estimación de pose permite la detección correcta de objetos y permite que los sistemas de manipulación puedan funcionar correctamente. La estimación de pose monocular además supone un ahorro de recursos de hardware al no necesitar de cámaras de profundidad, sensores LiDAR (*Light Detection And Ranging*) o varias cámaras para poder realizar la estimación.

Dentro del campo de realidad aumentada, los sistemas de estimación de pose permiten localizar la cámara respecto de un ambiente real o virtual. Con esta información es posible agregar diversos elementos visuales al ambiente de una forma natural y fluida, lo cual crea un ambiente inmersivo para el usuario.

La continuación natural de este proyecto sería desarrollar un sistema de refinamiento para poder realizar una comparación contra los sistemas de pose que poseen este tipo de sistemas. También sería muy útil desarrollar un marco de trabajo para poder realizar los modelos de objetos propios de una forma rápida e intuitiva para el usuario.

También sería muy útil realizar un análisis más amplio acerca de la capacidad de la red para aprender objetos más pequeños o con geometrías más complejas.

Bibliografía

[1] Arthur Louis Alaniz II y Christina Marianne Mantaring. "Real-Time Camera Pose Estimation for Virtual Reality Navigation". En: ().

[2] Herbert Bay, Tinne Tuytelaars y Luc Van Gool. "Surf: Speeded up robust features". En: *European conference on computer vision*. Springer. 2006, págs. 404-417.

[3] Catherine Capellen, Max Schwarz y Sven Behnke. "ConvPoseCNN: Dense Convolutional 6D Object Pose Estimation." En: (2019). URL: http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsarx&AN=edsarx.1912.07333&lang=es&site=eds-live.

[4] Bo Chen y col. "End-to-End Learnable Geometric Vision by Backpropagating PnP Optimization." En: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Pattern Recognition (CVPR), 2020 IEEE/CVF Conference on, CVPR* (2020), págs. 8097-8106. ISSN: 978-1-7281-7168-5. URL: http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.9156614&lang=es&site=eds-live.

[5] Alvaro Collet, Manuel Martinez y Siddhartha S Srinivasa. "The MOPED framework: Object recognition and pose estimation for manipulation". En: *The International Journal of Robotics Research* 30.10 (2011), págs. 1284-1306. DOI: 10.1177/0278364911401765. eprint: https://doi.org/10.1177/0278364911401765. URL: https://doi.org/10.1177/0278364911401765.

[6] Daniel DeTone, Tomasz Malisiewicz y Andrew Rabinovich. "SuperPoint: Self-Supervised Interest Point Detection and Description." En: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Computer Vision and Pattern Recognition Workshops (CVPRW), 2018 IEEE/CVF Conference on, CVPRW* (2018), págs. 337-33712. ISSN: 978-1-5386-6100-0. URL: http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.8575521&lang=es&site=eds-live.

[7] Martin A Fischler y Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". En: *Communications of the ACM* 24.6 (1981), págs. 381-395.

[8] Stefan Hinterstoisser y col. "Gradient response maps for real-time detection of textureless objects". En: *IEEE transactions on pattern analysis and machine intelligence* 34.5 (2011), págs. 876-888.

[9] Stefan Hinterstoisser y col. "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes". En: *Asian conference on computer vision*. Springer. 2012, págs. 548-562.

[10] Stefan Hinterstoisser y col. "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes". En: *2011 international conference on computer vision*. IEEE. 2011, págs. 858-865.

[11] Wadim Kehl y col. "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again". En: *Proceedings of the IEEE international conference on computer vision*. 2017, págs. 1521-1529.

[12] Vincent Lepetit, Francesc Moreno-Noguer y Pascal Fua. "Epnp: An accurate o (n) solution to the pnp problem". En: *International journal of computer vision* 81.2 (2009), pág. 155.

[13] Guosheng Lin y col. "Fast supervised hashing with decision trees for high-dimensional data". En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, págs. 1963-1970.

[14] David G Lowe. "Object recognition from local scale-invariant features". En: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee. 1999, págs. 1150-1157.

[15] Aiden Nibali y col. "Numerical coordinate regression with convolutional neural networks". En: *arXiv preprint arXiv:1801.07372* (2018).

[16] Sida Peng y col. "Pvnet: Pixel-wise voting network for 6dof pose estimation". En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, págs. 4561-4570.

[17] Mikael Persson y Klas Nordberg. "Lambda twist: An accurate fast robust perspective three point (P3P) solver". En: *Proceedings of the European conference on computer vision (ECCV)*. 2018, págs. 318-332.

[18] Ethan Rublee y col. "ORB: An efficient alternative to SIFT or SURF". En: *2011 International conference on computer vision*. Ieee. 2011, págs. 2564-2571.

[19] Paul-Edouard Sarlin y col. "SuperGlue: Learning Feature Matching With Graph Neural Networks." En: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Computer Vision and Pattern Recognition (CVPR), 2020 IEEE/CVF Conference on, CVPR* (2020), págs. 4937-4946. ISSN: 978-1-7281-7168-5. URL: `http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.9157489&lang=es&site=eds-live`.

[20] Torsten Sattler y col. "Understanding the limitations of cnn-based absolute camera pose regression". En: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, págs. 3302-3312.

[21] Christian Szegedy y col. "Inception-v4, inception-resnet and the impact of residual connections on learning". En: *Thirty-first AAAI conference on artificial intelligence*. 2017.

[22] Bugra Tekin, Sudipta N Sinha y Pascal Fua. "Real-time seamless single shot 6d object pose prediction". En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, págs. 292-301.

[23] Yu Xiang y col. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes." En: (2017). URL: http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsarx&AN=edsarx.1711.00199&lang=es&site=eds-live.

[24] Sergey Zakharov, Ivan Shugurov y Slobodan Ilic. "DPOD: 6D Pose Object Detector and Refiner." En: *2019 IEEE/CVF International Conference on Computer Vision (ICCV), Computer Vision (ICCV), 2019 IEEE/CVF International Conference on* (2019), págs. 1941-1950. ISSN: 978-1-7281-4803-8. URL: http://pbidi.unam.mx:8080/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.9010850&lang=es&site=eds-live.