Entrenamiento de IA para aplicación a robótica industrial: Generación de elementos mediante geometrías para entorno virtual

Mikel Hualde Otamendi[1], **Estibaliz Barberena Tabar**[1], **Xabier Iriarte Goñi**[1,2], **Oier Saldaña Barroso**[1]

[1]Departamento de Ingeniería, Universidad Pública de Navarra (UPNA), Campus Arrosadía, 31006 Pamplona, España
[2]Institute of Smart Cities (ISC), Universidad Pública de Navarra (UPNA), Campus Arrosadía, 31006 Pamplona, España

Email: [mikel.hualdeo, estibaliz.barberena, xabier.iriarte, oier.saldana]@unavarra.es

Resumen

La introducción de la Inteligencia Artificial y las Redes Neuronales en la industria resulta muy ventajosa, ya que posibilita la automatización de muchas tareas repetitivas que precisan de sentidos físicos. Las Redes Neuronales requieren ser entrenadas con datasets, que consisten en imágenes y datos numéricos de interés. No obstante, el desarrollo de técnicas de generación de datasets para componentes industriales se está realizando de manera independiente y aislada entre empresas, con sus consiguientes dificultades. Este artículo aporta una metodología sencilla y económica para la generación de datasets de objetos industriales que presenten cierta aleatoriedad en su disposición o forma, que ha sido llevada a la práctica en dos componentes distintos de dos empresas reales. La creación de estos datasets es muy versátil, pudiendo utilizarlos posteriormente para entrenar Redes Neuronales con distintos fines, como la generación de trayectorias para brazos robóticos o la Detección de Objetos, entre otros.

Palabras clave: Modelos 3D; Dataset; Red Neuronal; Detección de Objetos; Robótica Industrial.

Abstract

The use of Artificial Intelligence and Neural Networks in the industry is very advantageous, due to the fact that it enables the automation of many repetitive tasks that require human senses. Neural Networks are trained employing datasets, which consist of images and some data of interest. Nevertheless, the development of dataset generation techniques for industrial components is being carried out in an independent and isolated way among companies, with its consequent difficulties. This paper provides a simple and economical methodology for the generation of datasets of industrial objects that show some randomness in their layout or shape. This methodology has been put into practice in two different components belonging to two real companies. The creation of these datasets is very versatile, being able to use them to train different Neural Networks, such as the generation of trajectories for robotic arms or Object Detection, among others.

Keywords: 3D models; Dataset; Neural Networks; Object Detection; Industrial Robotics.

1. Introducción

El rápido avance de la robótica industrial ha posibilitado la automatización de gran parte de las operaciones más mecánicas y repetitivas de la industria. Sin embargo, siguen existiendo muchas operaciones repetitivas que requieren de sentidos intrínsecamente humanos, como la visión, que permiten al trabajador adaptarse a la aleatoriedad de ciertas tareas. Es el caso, por ejemplo, del asimiento de un objeto por una zona concreta, la soldadura de una ranura o la aplicación de pintura en una superficie específica. En estas situaciones, el objeto sobre el que se actúa puede estar dispuesto de formas distintas y aleatorias, y es necesario identificar correctamente la zona de interés.

El desarrollo exponencial de la Inteligencia Artificial (IA) está abriendo vías para la automatización de muchas de esas tareas, gracias a la identificación de

objetos mediante Visión Artificial y el entrenamiento de Redes Neuronales.

Dentro del mundo de la Visión Artificial, la Detección de Objetos (*Object Detection*) es precisamente eso: la identificación de objetos de interés dentro de una imagen.

Las Redes Neuronales requieren de un periodo de entrenamiento para aprender a identificar los objetos y puntos de interés. Este aprendizaje se lleva a cabo proporcionando a la Red Neuronal un conjunto de datos que consiste en imágenes de los objetos en cuestión y las coordenadas de los puntos de interés respecto de un sistema de referencia (o vectores directores de alguna curva, etc.). A este conjunto de datos se le denomina set de datos (dataset). La Red Neuronal aprenderá a asociar las imágenes con las coordenadas, y posteriormente será capaz de obtener estas últimas a partir de imágenes de entornos cambiantes. Realizar este proceso de aprendizaje en entornos reales es muy costoso, tanto en términos de tiempo como económicos. Así, resulta útil realizar una primera fase de aprendizaje en un entorno virtual, en el que es más sencillo simular entornos cambiantes con una gran variabilidad. Este artículo se va a centrar en la creación de modelos 3D y datasets para el entrenamiento de Redes Neuronales (tanto para Detección de Objetos como para alimentar la generación de trayectorias del artículo de Merino et al. mediante Imitation Learning [1]).

Actualmente este campo está siendo muy investigado y desarrollado, existiendo una gran cantidad de artículos, modelos 3D y datasets con este propósito. Sin embargo, estos avances han estado desacompasados: los datasets y los modelos de objetos comunes (*common-object datasets*) superan enormemente a los de objetos mecánicos y del ámbito industrial.

Tal y como afirman Sangpil Kim et al. [2], el escaso interés en crear un extenso dataset de componentes mecánicos e industriales está ralentizando el desarrollo y la evaluación de algoritmos de aprendizaje para la detección de objetos mecánicos. Esto responde principalmente a derechos de propiedad y de secreto industrial de las compañías.

Al no disponer de datasets accesibles, muchas empresas realizan el desarrollo de sus redes neuronales de una forma aislada e independiente, sin disponer de una base de datos común de la que poder partir. Las compañías, por lo tanto, disponen de diseños particulares y en muchas ocasiones confidenciales, por lo que se tiende a recurrir al escaneo 3D de las piezas con sus inconvenientes en costes, tamaños y volúmenes máximos que se pueden escanear [3,4]. Otra opción es la fotogrametría, que consiste en generar los modelos tridimensionales a partir de un conjunto de imágenes [5,6]. Ambos procedimientos dan como resultado nubes de puntos, que pueden ser reconstruidas en modelos 3D (mallas o *mesh*es, más concretamente).

No obstante, estos dos métodos tan extendidos no resultan adecuados en el caso de tener que modelar piezas o componentes que tengan cierta variabilidad o aleatoriedad, ya que haría falta realizar el proceso de escaneo o de toma de fotografías repetidas veces.

La diferencia entre un modelo mallado (como un archivo STL, por ejemplo) y un modelo sólido (también llamados modelos CAD) es que los primeros son una descomposición en polígonos de la superficie de la pieza, mientras que los últimos describen una secuencia de pasos con la que representar las geometrías vectorialmente (con lo que pueden ser fácilmente modificados).

La contribución de este trabajo, por lo tanto, se fundamenta en la exposición de un método sencillo y económico para modelar piezas, ensamblajes y/o productos manufactureros mediante geometrías y parámetros que presenten una cierta aleatoriedad, para obtener imágenes de modelos 3D con disposiciones aleatorias en entornos variables. Este modelado servirá para generar escenarios virtuales de entrenamiento de redes neuronales para la automatización de operaciones mediante robots industriales. Para ello, se van a crear modelos sólidos con ciertas dimensiones parametrizadas, de modo que sea posible variarlas aleatoriamente en un rango deseado.

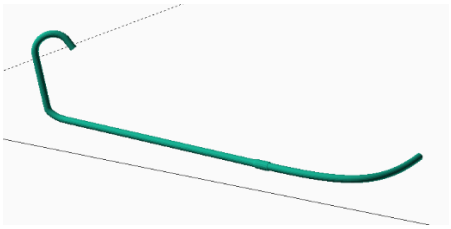
2. Metodología: Procedimiento general

El proceso de generación de modelos 3D y datasets se ha llevado a cabo con dos componentes reales empleados en la industria: un cable enrollado y un tubo, cada uno perteneciente a un producto final totalmente distinto, de empresas de sectores muy diferentes. El primero es para el ABS de un vehículo, y el segundo es parte de un frigorífico. Esto muestra la gran versatilidad de este método, que es aplicable a una gran variedad de objetos, siempre que estos posean algún atributo que varíe aleatoriamente.

Figura 1. Fotografía del cable de ABS real.
Fuente: SKF Española, S.A.

Por un lado, el cable de ABS puede apreciarse en la Figura 1. La parte flexible está enrollada en su zona central, teniendo cierta variabilidad en el radio de dicha circunferencia así como en la disposición de sus extremos. En estos últimos presenta unos conectores rígidos, cuya posición y orientación podrá variar, pero no así su forma. En esta aplicación el robot se dirigirá al cable y lo asirá por los conectores.

Por otro lado, el tubo se compone de una parte rígida y de otra flexible. La parte rígida está soldada al compresor por uno de sus extremos (zona izquierda de la Figura 2.a), y tiene un ensanchamiento de diámetro en el extremo opuesto (zona central de la Figura 2.a). Este último se une por soldadura a un tubo flexible que gira 90º en un plano prácticamente horizontal (zona derecha de la Figura 2.a). El robot deberá realizar estas dos uniones soldadas.



(a)

(b)

Figura 2. Imágenes 3D del tubo aislado (a) y del frigorífico completo con el tubo coloreado en verde (b).
Fuente: BSH Electrodomésticos España.

Se van a emplear estos dos objetos para ilustrar la secuencia de pasos que se ha seguido en esta metodología.

Para generar las geometrías de forma paramétrica e introducir cierta aleatoriedad, se han utilizado distintas herramientas informáticas, cumpliendo cada una con una función específica:

- *Openscad*: aplicación CAD de representación de modelos 3D, de código abierto, cuyas operaciones de representación de piezas se llevan a cabo mediante código. Con ella se crearán los modelos sólidos.
- *Gazebo*: herramienta de simulación enfocada a aplicaciones robóticas, personalizable mediante interfaz gráfica o código. En este entorno de simulación se tomarán las imágenes.
- *Matlab:* aplicación que realizará un bucle de iteraciones, es decir, un bucle de distintas configuraciones del objeto en cuestión. Además, efectuará los cálculos, se encargará de la comunicación entre las tres aplicaciones y organizará todos los datos generados, dando lugar al dataset.

Estos serían los pasos necesarios para generar el dataset:

1) Representación esquemática de la pieza mediante geometrías más sencillas y parametrizables
2) Representación 3D del objeto y creación del entorno de simulación
3) Creación de un script para dar variabilidad
4) Almacenamiento de las imágenes
5) Generación del dataset

A continuación se detalla cada uno de los pasos mencionados.

2.1. Representación esquemática de la pieza mediante geometrías más sencillas y parametrizables

Para comenzar, se deben escoger geometrías elementales con las que poder representar la pieza, buscando un compromiso entre la sencillez y la similitud con la pieza real. Se establecerán entonces las dimensiones de dichas geometrías, definiendo parámetros para las variables y el rango de variabilidad de los mismos.

Además, se deberá reflexionar sobre qué componentes son siempre iguales y no merece la pena parametrizar, ya que este método permite unir modelos 3D (e.g., STEP) o meshes (e.g., STL) al modelo 3D parametrizado. Esto es especialmente interesante cuando se tienen componentes o partes muy complejas

de asemejar mediante geometrías elementales y que no van a variar. Esto sucede con los dos ejemplos analizados, en concreto con los conectores del cable (véanse los extremos del cable de la Figura 1) y la mayoría de componentes del frigorífico (véase Figura 2.b).

2.2. Representación 3D del objeto y creación del entorno de simulación

El siguiente paso consiste en representar el objeto parametrizado mediante una aplicación CAD de generación de modelos 3D. Como se ha mencionado anteriormente, se ha optado por la plataforma *Openscad*, en lugar de realizar diseños a través de una interfaz gráfica. Esto permite definir variables dentro del script de *Openscad*, que más adelante se modificarán aleatoriamente y de forma externa.

Por lo tanto, en este paso se escogen unos valores concretos para los parámetros, es decir, se particulariza para un caso concreto. Una vez creado el sólido, se puede exportar en diversos formatos (STL entre ellos), y se importa en la plataforma *Gazebo*, que permite visualizar objetos 3D en entornos muy variados (pudiendo modificar una gran cantidad de parámetros visuales, como la luminosidad, la dirección de las luces, los colores, añadir otros muchos objetos, etc.). Se ha escogido *Gazebo* porque permite la inserción y control de brazos robóticos, lo cual es interesante para el objetivo de alimentar el entrenamiento de redes neuronales en el que se enmarca este artículo. Además, en *Gazebo* se pueden colocar cámaras en distintas ubicaciones, con las cuales se tomarán las imágenes para la creación del dataset.

Así pues, se creará un archivo de *Gazebo* (llamado *world* o entorno de *Gazebo*) en el cual se importará el modelo sólido creado en *Openscad*.

2.3. Creación de un script para dar variabilidad

Una vez creado el modelo en *Openscad*, se crea un programa en *Matlab* que le pueda transferir los valores de los parámetros; es decir, el archivo de *Openscad* será una mera "plantilla" donde *Matlab* escribirá los valores concretos que adquieran los parámetros en cada iteración. De esta forma, desde *Matlab* se puede dar un valor a cada parámetro, y *Matlab* se encargará de generar el modelo y el STL en *Openscad* con esos valores concretos.

Tras haber realizado la interconexión entre *Matlab* y *Openscad*, resulta sencillo integrar el programa de *Matlab* en un bucle en el cual los valores de los parámetros varíen de forma aleatoria dentro de un rango especificado. De esta manera se puede introducir la variabilidad debida a las tolerancias de la pieza estudiada.

2.4. Almacenamiento de las imágenes

Llegados a este punto, es posible crear modelos tridimensionales (en formato sólido y STL) aleatorios. Sin embargo, no se han obtenido todavía imágenes de los modelos de forma automatizada. Para ello, se ha creado una "plantilla" del mundo de *Gazebo* (al igual que con el archivo de *Openscad*), sobre la cual el programa de *Matlab* será capaz de realizar modificaciones.

En este caso, se deberá modificar la ruta de almacenamiento de las imágenes que toman las cámaras, así como sus nombres, de modo que resulte sencillo ordenarlas después en el dataset.

La "plantilla" de *Gazebo* realizada en el apartado anterior puede volverse mucho más versátil si además de la ruta de almacenamiento de las imágenes se modifican otras opciones del entorno de *Gazebo*. Como ejemplo, *Gazebo* permite modificar muchas variantes ambientales de la simulación, como los tipos de luces, sus orientaciones y colores, el fondo, los objetos del entorno, el terreno o suelo sobre el que colocarlos, etc. Se viene comprobando que la variación de condiciones ambientales influye positivamente en las etapas de entrenamiento de las Redes Neuronales [7,8], adaptándose posteriormente al mundo real sin un salto (conocido como *gap*) tan grande.

2.5. Generación del dataset

Finalmente, se creará un documento de texto en el cual se escriban de forma ordenada (matricial, por ejemplo) los datos requeridos para el entrenamiento del algoritmo de Inteligencia Artificial: las coordenadas de los puntos de interés, los vectores directores de algunos objetos, etc. Esto depende de qué operación se quiera efectuar y en qué zona o punto del objeto.

Cada fila de esta matriz de datos corresponde a una configuración nueva y aleatoria del objeto en cuestión, y por lo tanto tendrá una imagen asociada a ella. Es recomendable emplear un nombre de archivo similar con el que poder asociarlas posteriormente, o directamente almacenarlas juntas en un directorio individual.

Por lo tanto, una vez obtenidos tanto la matriz de datos como las imágenes**,** se dispone ya del dataset al que llamará el algoritmo de entrenamiento de la Red Neuronal.

3. Ejemplos de aplicación

Tras la exposición de la metodología empleada, se va a ilustrar el procedimiento con los dos objetos mencionados anteriormente: un cable para el ABS de un vehículo y un tubo perteneciente a un frigorífico.

3.1. Esquema y generación de geometría del cable

En el caso del cable, como se ha visto en la Figura 1, el modelo real está enrollado sobre sí mismo y tiene dos conectores con unas características muy concretas en sus extremos. Este componente se ha representado como una circunferencia con dos líneas tangentes y dos conectores en los extremos (véase Figura 3). El radio puede variar en un cierto rango, por lo que las líneas tangentes pueden verse acortadas o alargadas, y pueden ser de distinta longitud. En cuanto a los conectores, conservarán su geometría real, ya que se han empleado STLs reales.

Los parámetros establecidos son los siguientes:

- el radio de la circunferencia (R),
- el ángulo de giro (Φ),
- el ángulo de apertura (α),
- las longitudes de las tangentes (L_1 y L_3);

y se han realizado los cálculos pertinentes para que la longitud total del cable se mantenga constante.

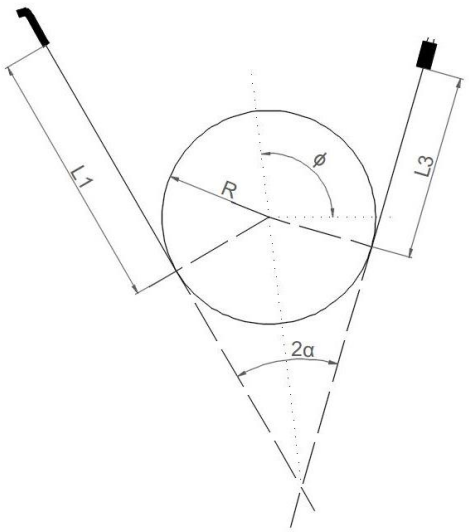


Figura 3. Representación esquemática y parametrizada del cable.

La representación del cable en *Openscad* se ha realizado sobre un plano, y consiste en la unión de tres sólidos extruidos (uno circular y dos lineales), trasladados y rotados según los ángulos de giro y de apertura. En los extremos libres de los sólidos lineales se han añadido los dos conectores, importándolos como archivos STL. La posición de todo el conjunto es aleatoria.

Openscad no permite la importación de archivos STEP, pero sí archivos STL, por lo que es necesario convertir los conectores a dicho formato. Posteriormente, se importan en *Openscad* y se realiza la unión con el cable con los desplazamientos y rotaciones necesarias, tras lo cual se desea exportar un archivo STL del conjunto. No obstante, inicialmente *Openscad* no era capaz de crear dicho STL, alegando que el mallado no estaba debidamente cerrado. Esto ocurría debido a que los conectores estaban compuestos por muchos detalles y algunos sólidos independientes en su interior, todos muy próximos físicamente, y en el momento de creación del STL a partir del STEP se generó un mallado no-cerrado (también llamado *non-manifold* o *not watertight*), es decir, existían mallados no cerrados o volúmenes separados que compartían únicamente un vértice o una arista. En este caso, se emplearon herramientas como *Meshlab* y *Solidworks* para eliminar ciertos detalles de pequeño tamaño y convertir en macizos los conectores. Esta sencilla solución dio resultado para el cable, pero no para el tubo (como se verá en el apartado 3.2). Un ejemplo de la representación del cable en *Openscad* puede apreciarse en la Figura 4.

Figura 4. Representación 3D de una configuración aleatoria del cable en *Openscad*.

En *Gazebo*, el cable se ha colocado en posiciones aleatorias sobre un bloque con las dimensiones de una mesa de trabajo. Se ha aleatorizado la iluminación (su posición, su orientación y sus colores), y también los colores tanto del cable como de la mesa, que son en gran parte función de la luz incidente. Se han tomado imágenes del entorno de *Gazebo* desde dos cámaras colocadas en posiciones distintas, para poder aportar la sensación de profundidad a la Red Neuronal. La Figura 5 muestra varias iteraciones del cable.

En esta aplicación el robot deberá asir el cable por la zona central de los conectores, de modo que los datos de interés para el dataset son las coordenadas de estos dos puntos y los vectores directores que indican la orientación de los conectores. Estos datos se han almacenado en un documento de texto junto a las imágenes, como se ha especificado en el apartado 2.5, dando lugar así al dataset.

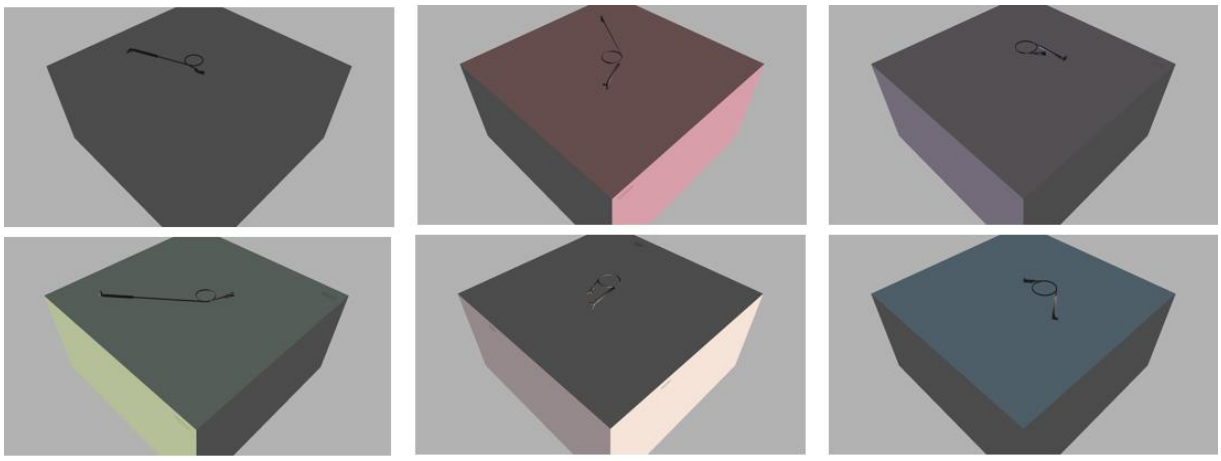


Figura 5. Ejemplo de seis iteraciones para el caso del cable. La disposición del cable, así como la iluminación y los colores, varían en cada iteración.

3.2. Esquema y generación de geometría del tubo

La geometría del tubo ha resultado más compleja de modelar que la del cable. Primeramente, el tubo (véase Figura 2.a) se ha dividido en tramos rectos y curvos. Los puntos de unión entre dichos tramos serán los que tengan variabilidad en el espacio, manteniendo siempre constante la longitud de los tramos rectos. La unión entre el tubo y el resto de la nevera se realiza posteriormente en *Gazebo* (uniendo los dos sólidos en formato STL).

Atendiendo a la Figura 6, se observa que los tramos Q_1-Q_2, Q_3-Q_4 y Q_5-Q_6 son rectos, y que los tramos Q_0-Q_1, Q_2-Q_3 y Q_4-Q_5 son curvos. Los puntos Q_0 y Q_6 son fijos, mientras que el resto de puntos tienen cierta variabilidad espacial.

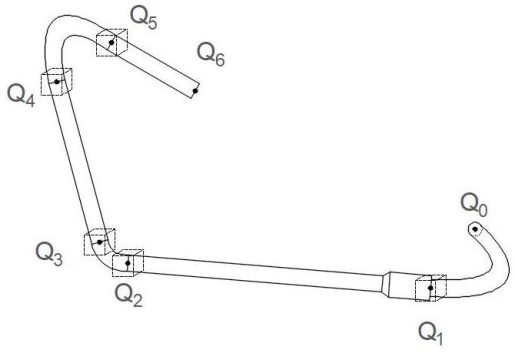


Figura 6. Representación esquemática y parametrizada del tubo.

El procedimiento se resume en la siguiente secuencia de pasos, que se explicarán posteriormente:

- Se colocan los puntos Q_i aleatoriamente en el interior de sus respectivos cubos tridimensionales
- Se generan los vectores directores desde Q_i a Q_{i+1} (tramo recto)
- Se establecen los nuevos puntos Q_i (criterio de longitud constante en tramos rectos)
- Se generan curvas de Bézier de Q_i a Q_{i+1} (tramo curvo)
- Se realiza la extrusión, recta o curva en función de cada tramo

Inicialmente, Matlab asigna valores aleatorios a los puntos (Q_i), cada uno de ellos dentro de un cubo tridimensional que será función de las tolerancias que pueda presentar dicho punto. En los tramos rectos se unen los puntos mediante una recta y obtienen así los vectores directores de los tramos rectos ($\mathbf{u}_i$ y $\mathbf{u}_{i+1}$). Además, se ajustan los dos puntos extremos de cada tramo recto de forma que la distancia entre ellos sea constante (L_i, la longitud de dicho tramo). La Figura 7 muestra este procedimiento, donde los puntos de mayor grosor representan los puntos aleatorios iniciales, mientras que los puntos de menor grosor representan los puntos recolocados, que tienen entre ellos una distancia L_i.

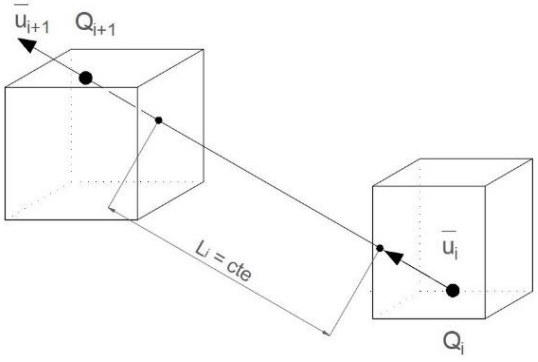


Figura 7. Colocación aleatoria y posterior colocación definitiva de los puntos extremos de los tramos rectos, manteniendo siempre constante la distancia entre ellos.

Como los tramos rectos y curvos están intercalados, estos últimos ya están totalmente definidos por los puntos y los vectores directores (véase Figura 8); así que se emplea una curva de Bézier de tercer orden. Se tiene libertad para colocar los dos puntos de control intermedios, por lo que se establece un criterio de decisión. En este caso, los puntos de control intermedios son los dos puntos que están a mínima distancia entre las dos rectas que forman los vectores directores de Q_i y Q_{i+1}, representados como P_1 y P_2 en la Figura 8.b. Teniendo los cuatro puntos de control de cada tramo curvo, se pueden calcular las trayectorias descritas por las curvas de Bézier de tercer orden.

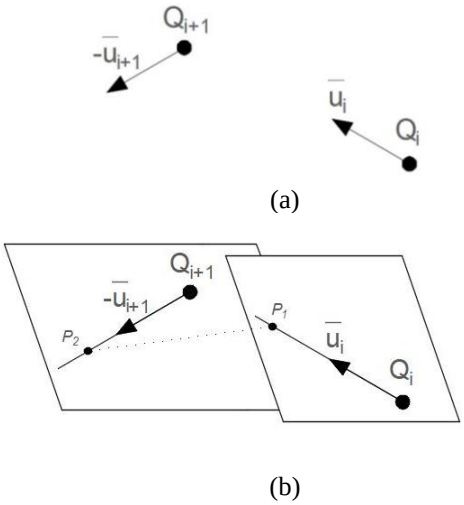


Figura 8. (a) Los puntos extremos de los tramos curvos y sus vectores directores han quedado definidos cuando se establecen los tramos rectos. (b) Se obtienen los puntos de mínima distancia entre rectas, que serán los puntos de control intermedios.

Una vez obtenidos todos los puntos Q_i y las trayectorias discretizadas de los tramos curvos (una serie de puntos pertenecientes a estos tramos), se transfieren estos datos a la "plantilla" de *Openscad*. La representación en *Openscad* puede hacerse creando un programa propio que represente tramos rectos y curvas de Bézier o utilizando funciones creadas por otros usuarios de *Openscad*. El procedimiento ha consistido en representar los tramos rectos mediante extrusión entre los dos puntos extremos, y paralelamente representar los tramos curvos como un barrido de una sección constante a lo largo de una trayectoria curva.

Inicialmente se intentó realizar la unión tubo-frigorífico en el propio *Openscad*, y exportar todo ello como un único STL. Sin embargo, el frigorífico del que se disponía estaba en formato STEP, de modo que primeramente se debía eliminar el tubo, y posteriormente convertir el resto del frigorífico a formato STL para que *Openscad* pudiese importarlo. Tras ello, y una vez unidos en *Openscad* el frigorífico y el tubo (este último creado íntegramente en *Openscad*), se procedía a exportar el objeto completo como un único STL. No obstante, al igual que ocurría con el cable, *Openscad* no era capaz de crear dicho STL, debido a la presencia de un sólido no-cerrado (*non-manifold*). Así como en el caso del cable el problema se resolvió de manera sencilla, el frigorífico está compuesto por muchos más sólidos, y todos muy cerca los unos de los otros, por lo que "simplificar" la geometría del frigorífico era mucho más complicado. Por ello, la unión entre tubo y frigorífico se realizó directamente en *Gazebo*.

Finalmente, en *Gazebo* se ha realizado una "plantilla", al igual que con el cable, y se ha añadido el frigorífico completo (sin el tubo y en formato STL), realizando los desplazamientos horizontales y giros necesarios para colocarlos en la posición relativa correcta. Se han aleatorizado también ciertos aspectos ambientales y de aspecto, y se han tomado las imágenes desde dos cámaras distintas. Algunos ejemplos se muestran en la Figura 9.

En este caso, los puntos Q_1 y Q_6 serán soldados por un brazo robótico, por lo que tanto su posición (coordenadas) como su orientación (vectores directores $\mathbf{u}_1$ y $\mathbf{u}_6$) se almacenarán en un documento de texto. Este último, junto a las imágenes de las cámaras, formarán el dataset.

4. Resultados y discusión

Mediante esta metodología se obtienen datasets para componentes que presentan cierta variabilidad. Estos datasets consisten en imágenes y datos (coordenadas de algún punto, vectores directores, etc.), agrupados en directorios para cada configuración del elemento modelado. Los datos están escritos en archivos de texto, organizados por columnas y separados por espacios (aunque podrían emplearse símbolos también).

Los dos objetos modelados en este artículo (el cable y el tubo) se enmarcan en el proyecto ImitAI, donde serán empleadas para el aprendizaje de trayectorias para brazos robóticos [1]. No obstante, este método permite la creación de datasets para aplicaciones muy variadas, como por ejemplo la Detección de Objetos.

En cuanto al tiempo requerido para generar los datasets, en el caso del cable la duración es de 230 segundos para 10 iteraciones, y de 274 segundos para 10 iteraciones del tubo. Todo ello se ha realizado en un ordenador con un procesador Intel Core i5-9500 CPU de 3.00 GHz, una RAM de 32 GB y una tarjeta gráfica de Mesa Intel UHD Graphics 630 (CFL GT2).

Las herramientas utilizadas se han ido escogiendo a lo largo del desarrollo del método, a raíz de diversas problemáticas que fueron apareciendo. *Matlab* es la

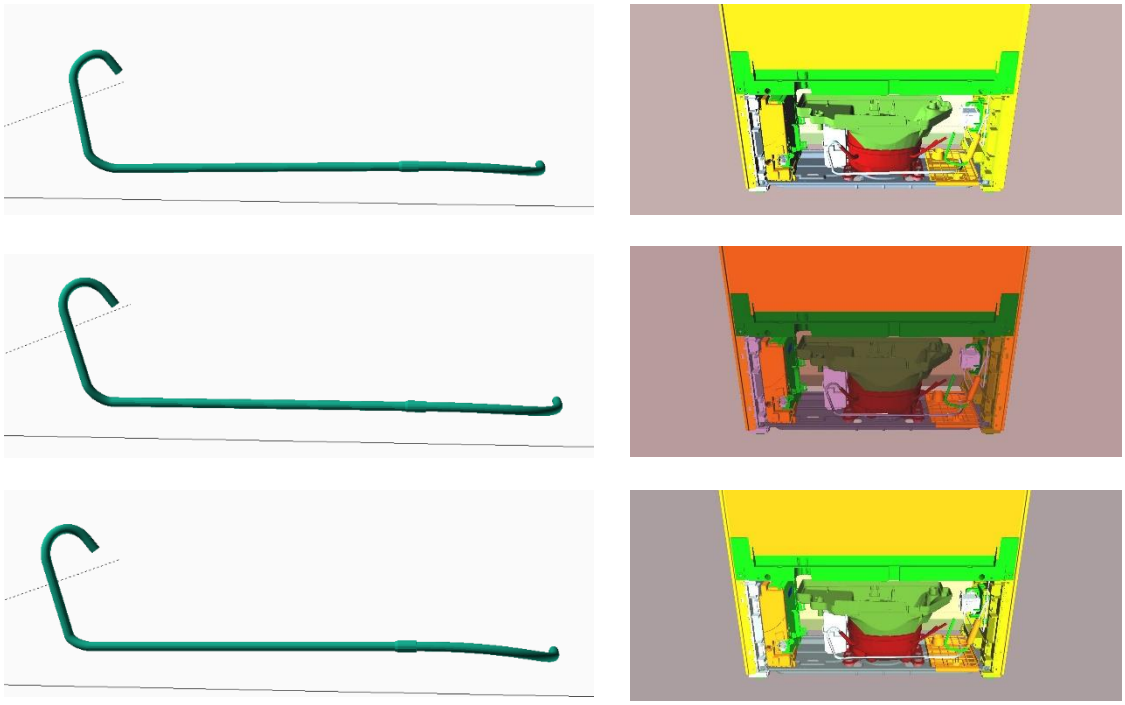


Figura 9. Ejemplo de cuatro iteraciones para el caso del tubo (aislado y con el frigorífico). La configuración del tubo, así como la iluminación y los colores, varían en cada iteración.

herramienta principal, que calcula, que articula e integra todas las operaciones, además de ser aquella que realiza un bucle de iteraciones. Se pueden emplear otras aplicaciones similares, como *Python*, *Octave* o *Scilab* y llegar a un resultado similar.

En cuanto a la creación de modelos tridimensionales, inicialmente se consideró la posibilidad de hacer uso de herramientas con interfaz gráfica (como *Solidworks*), pero rápidamente se reconoció la facilidad de parametrizar variables en *Openscad* y de modificar sus archivos mediante *Matlab*. Sin embargo, hubo ciertas complicaciones al importar archivos STL que no estuviesen correctamente cerrados y tratar de juntarlos con nuevos sólidos. Esto se solucionó "simplificando" sólidos (en el caso del cable) o directamente realizando las uniones en *Gazebo* (en el ejemplo del tubo).

Por último, el uso de la herramienta *Gazebo* se fundamenta en dos razones: por un lado, la conveniencia de este simulador en cuanto a poder añadir y controlar brazos robóticos, posibilitando así la toma de imágenes tanto para aplicaciones de Detección de Objetos como de generación de trayectorias para robots; por otro lado, la gran customización de entornos que posibilita *Gazebo*.

Las herramientas que se han utilizado, sin embargo, no son las únicas con las que se podría llevar a cabo este método. Se podrían emplear otras aplicaciones y obtener resultados muy similares.

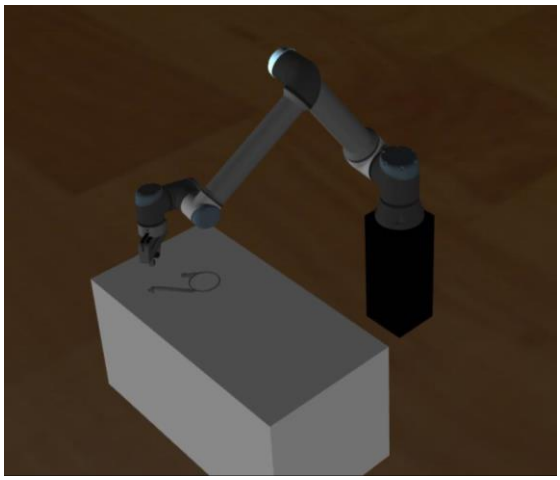


Figura 10. Imagen tomada en *Gazebo*, donde se muestra una disposición aleatoria del cable, junto al brazo robótico UR10.

5. Conclusiones

Este artículo presenta un método sencillo, versátil y económico para la generación de datasets de piezas, ensamblajes y/o productos industriales que presenten cierta variabilidad en su disposición o forma. Estos datasets serán parte del entrenamiento de redes neuronales que podrán tener diversos objetivos, siendo el más conocido la Detección de Objetos.

Este método permite modelar objetos y componentes que presenten cierta aleatoriedad, parametrizando las

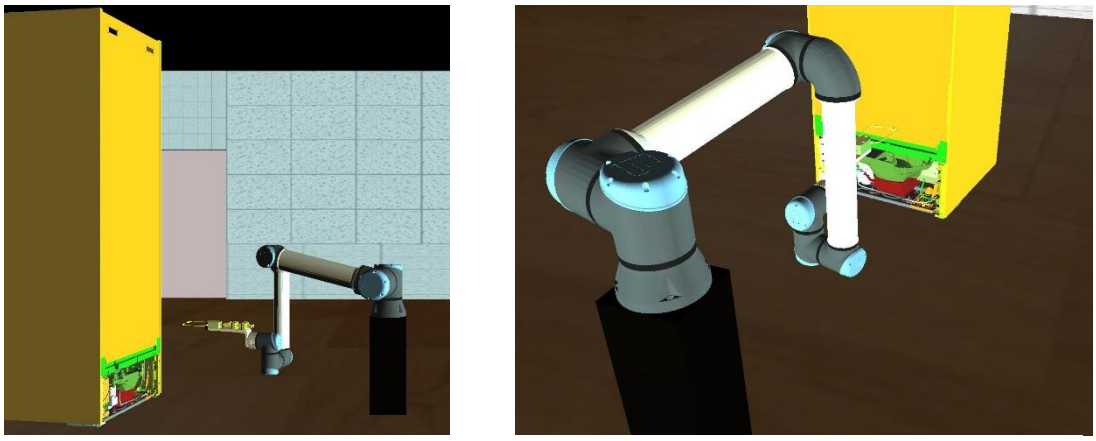


Figura 11. Imágenes tomadas en *Gazebo* por dos cámaras colocadas en distintas posiciones, donde se muestra una disposición aleatoria del tubo, junto al brazo robótico UR10.

dimensiones variables y creando modelos 3D también variables. Se introducen posteriormente en entornos de simulación cambiantes, donde se toman imágenes del objeto, y en caso de ser necesario también del robot. Finalmente, se almacenan tanto estas imágenes como los datos de interés, obteniendo así un set de datos.

Los casos prácticos que ilustran este artículo son un cable para el ABS de un vehículo y un tubo perteneciente a un frigorífico. Ambos objetos pueden presentar cierta aleatoriedad en su disposición o forma, por lo que se ha estudiado cada caso particular para ser modelado con las geometrías más adecuadas y para parametrizar las dimensiones que varíen. Los modelos 3D generados siguiendo esta metodología, aunque podrían utilizarse para ser detectados por Visión Artificial, se van a emplear como detalles visuales en imágenes que muestren las trayectorias ideales que debería seguir un brazo robótico (véanse la Figura 10 y la Figura 11 como ejemplos de estas imágenes). Estas imágenes servirán para entrenar la red neuronal de generación de trayectorias, que guiará al robot en sus tareas de asimiento del cable y soldadura del tubo. Todo ello se enmarca en el proyecto ImitAI para robots industriales de precisión con Inteligencia Artificial basada en aprendizaje por Imitación.

6. Agradecimientos

Este trabajo fue financiado por la "Convocatoria de ayudas a proyectos estratégico de I+D del Gobierno de Navarra", bajo el proyecto *Robots Industriales de Precisión con inteligencia Artificial Basada en Aprendizaje por Imitación (ImitAI)* con Ref. 0011-1411-2021-000023.

7. Referencias

[1] M. Merino Olagüe, J. Ibarrola Chamizo, J. Aginaga García y M. Hualde Otamendi, «Entrenamiento de IA para aplicación a robótica industrial: Generación de trayectorias para ajuste de parámetros de red neuronal,» *XV Congreso Iberoamericano de Ingeniería Mecánica,* 2022.

[2] S. Kim, H. Chi, X. Hu, Q. Huang y K. Ramani, «A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks,» *Computer Vision – ECCV 2020,* pp. 175-191, 2020.

[3] M. McGough, «Generating 3D Models with PolyGen and PyTorch,» Towards Data Science, 29 Octubre 2020. [En línea]. Disponible: https://towardsdatascience.com/generating-3d-models-with-polygen-and-pytorch-4895f3f61a2e. [Último acceso: 30 Junio 2022].

[4] A. M. Cretu, E. M. Petriu y G. G. Patry, «Neural network architecture for 3D object representation,» *Proceedings 2nd IEEE International Workshop on Haptic, Audio and Visual Environments and their Applications,* pp. 31-36, 2003.

[5] F. Remondino y S. El-Hakim, «Image-based 3D modelling: A review,» *The Photogrammetric Record 21,* vol. 21, nº 115, pp. 269-291, 2006.

[6] T. Luhmann, «Close range photogrammetry for industrial applications,» *ISPRS Journal of Photogrammetry and Remote Sensing,* nº 65, pp. 558-569, 2010.

[7] Y. Rao, B. Liu, Y. Wei, J. Lu y C. J. Hsieh, «RandomRooms: Unsupervised Pre-training from Synthetic Shapes and Randomized Layouts for 3D Object Detection,» *2021 IEEE/CVF International Conference on Computer Vision (ICCV),* pp. 3263-3272, 2021.

[8] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba y P. Abbeel, «Domain randomization for transferring deep neural networks from simulation to the real world,» *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 23-30, Septiembre 2017.