electronics

MDPI

Article

Development of a Virtual Environment for Rapid Generation of Synthetic Training Images for Artificial Intelligence Object Recognition

Chenyu Wang [1,2], Lawrence Tinsley [1] and Barmak Honarvar Shakibaei Asli [1,*]

[1] Centre for Life-Cycle Engineering and Management, Faculty of Engineering and Applied Sciences, Cranfield University, Bedford MK43 0AL, UK; chenyu.wang.370@cranfield.ac.uk (C.W.); l.tinsley@cranfield.ac.uk (L.T.)
[2] College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
* Correspondence: barmak@cranfield.ac.uk

Abstract: In the field of machine learning and computer vision, the lack of annotated datasets is a major challenge for model development and accuracy improvement. Synthetic data generation addresses this issue by providing large, diverse, and accurately annotated datasets, thereby enhancing model training and validation. This study presents a Unity-based virtual environment that utilises the Unity Perception package to generate high-quality datasets. First, high-precision 3D (Three-Dimensional) models are created using a 3D structured light scanner, with textures processed to remove specular reflections. These models are then imported into Unity to generate diverse and accurately annotated synthetic datasets. The experimental results indicate that object recognition models trained with synthetic data achieve a high rate of performance on real images, validating the effectiveness of synthetic data in improving model generalisation and application performance. Monocular distance measurement verification shows that the synthetic data closely matches real-world physical scales, confirming its visual realism and physical accuracy.

Keywords: synthetic data; game engine; target recognition; virtual environment

check for updates

Citation: Wang, C.; Tinsley, L.; Honarvar Shakibaei Asli, B. Development of a Virtual Environment for Rapid Generation of Synthetic Training Images for Artificial Intelligence Object Recognition. *Electronics* **2024**, *13*, 4740. https://doi.org/10.3390/electronics13234740

Academic Editors: Muchao Ye and Pan He

Received: 16 September 2024
Revised: 11 November 2024
Accepted: 27 November 2024
Published: 29 November 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the increasing prevalence of machine learning in robotics, the demand for data annotation has also rapidly increased. High-quality data have been proven to be the key driver of progress in computer vision, but it is also the primary limiting factor [1]. Particularly in the fields of semantic scene understanding and object detection, rich and accurate data are crucial for improving algorithm performance. Detailed annotation and labelling allow robots to better recognise and interpret complex environments and objects, thereby improving their autonomy and intelligence. This growing demand has also driven the development of new data annotation technologies and tools, further promoting innovation and application in robotics. Currently, there are several ways to efficiently label large amounts of training data. For example, Bryan C. Russell et al. present an image annotation tool based on polygonal and free-form annotations that supports online collaboration and text-based labelling, enabling open sharing of image annotation data [2]. Luis von Ahn and Laura Dabbish solve the image annotation problem by designing fun online games that allow players to provide meaningful labels for images in an entertaining way [3]. However, these methods are inherently limited by the human resources required for manual labelling or supervision. They also fail to address the difficulty and high cost of accessing raw data, mainly due to the diversity and fragmentation of data, quality issues, technical and tool limitations, privacy and security requirements, high access costs, and the frequency and complexity of data updates. Data come from multiple sources and in different formats,

requiring significant time and technology to process, while privacy protection and security risks add to the difficulty of access. The investment in advanced tools and specialists also keeps overall costs high.

The application of synthetic data has shown promising results, effectively addressing issues such as data scarcity and high annotation costs. By utilising advanced simulation environments and algorithms, synthetic data generation enables the creation of high-quality, scalable, and cost-effective datasets. This approach overcomes the constraints of manual labelling by automating the generation of diverse, richly annotated data. With the development of game engines such as Unity and Unreal Engine (UE), they have become powerful platforms for synthetic data. With the help of these engines, researchers and developers can build highly realistic virtual environments and simulate a variety of complex scenarios to generate a large amount of accurately labelled synthetic data. These engines not only provide powerful graphic rendering capabilities, but also support physical simulation, lighting calculations, and many other features that make the data generated more realistic and believable. In addition, the flexibility and ease of use of the game engines allow researchers to quickly iterate and customise the scenarios to meet the needs of different studies and applications.

This study aims to create a virtual environment for generating high-quality synthetic data to supplement real-world datasets, particularly when certain object data are scarce or difficult to obtain. The virtual environment supports the import and preprocessing of 3D models, allowing for realistic lighting and shadow effects. This enhances the quality and utility of the synthetic data, improving model training and object recognition accuracy. In addition, this approach provides researchers with a diverse option that complements existing synthetic data engines and datasets, offering additional support and flexibility for specific needs and application scenarios.

2. Literature Review

Synthesising data using virtual environments involves generating new, controlled samples based on prior knowledge. Specifically, it refers to obtaining particular images, videos, and their annotation information through simulated scenarios. This process is generally carried out using game engines like Unreal, Blender, and Unity to create synthetic data. Neural network techniques are then employed to bridge the gap in data distribution between synthetic and real data.

2.1. The Application of Virtual Environments

The use of Unity or Unreal Engine (UE) to build synthetic data engines has become a very popular trend, attracting the attention and contributions of many researchers. These engines are capable of generating virtual environments with a high degree of realism to support diverse visual tasks such as semantic segmentation, object detection, person re-identification, and autonomous driving. Various Unity- and UE-based virtual environments and datasets continue to emerge, providing rich labelled data and scene simulations for computer vision and deep learning research. However, these virtual environments also have certain limitations in terms of diversity and scene coverage, which affect the generalisation ability of models in the real world. Table 1 presents the virtual environments and datasets currently created using game engines (Unity and Unreal Engine).

However, both UE and Unity have a certain technical threshold, and researchers do not have a lot of time to learn to create high-quality models. So the use of large-scale online games as virtual environments has become a popular trend. These games usually have realistic models and rich scene content. For example, in recent years, a large number of researchers have used the game GTA-V for their own developmental research. Table 2 lists the publicly available datasets developed using GTA-V in recent years. GTA-V's vehicles, characters, buildings, and even the sky's colour are rendered with impressive realism, closely resembling reality. The game also showcases its rendering capabilities under various weather and lighting conditions, further enhancing the visual authenticity. Despite the fact that GTA-V's visuals are

improving and coming closer and closer to the real world, as a game it still has limitations in its scene layout and visual style that restrict its ability to generalise and be applicable in real-life scenarios. Specific limitations are as follows:

1. Because all the data are based on the game GTA-V, although its visual effects are coming closer to reality with the updates, as a game it still has its own unique colour scheme, object texture, and composition. Models trained under this 'game aesthetic' may not be able to adapt to the diversity of real scenes.
2. The scene layout in the game mainly focuses on urban environments, with insufficient coverage of rural, industrial, and natural scenes. This limitation leads to poor performance of the model when applied to other types of scenes, which affects its range of practical applications.

Table 1. Datasets/engine developed using Unity and Unreal Engine.

Author and Year	Engine	Dataset/Engine Name	Task	Limitations
German Ros et al. [4]	Unity	SYNTHIA	Semantic segmentation	Primarily focused on urban environments; lacking diversity in other scenes, limiting generalisation to the real world.
Adrien Gaidon et al. [5]	Unity	N/A	Object detection, tracking, scene and instance segmentation, depth, optical flow	Synthetic data still present a "domain gap" with real data, potentially affecting model performance in real-world applications.
Alexey Dosovitskiy et al. [6]	Unreal Engine	CARLA	Autonomous driving simulation	Virtual environment mainly includes urban and highway scenes; lacking rural or complex road conditions, limiting generalizability.
Xiaoxiao et al. [7]	Unity	PersonX	Pedestrian re-identification	Limited pedestrian models and scenes; insufficient to capture diverse pedestrian characteristics and lighting in real environments.
Manolis et al. [8]	Unity	MINOS	Indoor navigation	Constrained to indoor environments, limiting its applicability to outdoor or more complex multi-scene tasks.
Slawomir Bak et al. [9]	Unity	SyRI	Person re-identification	Focused solely on lighting variations; lacking other factors affecting re-identification, such as occlusion and diverse pedestrian poses.

Table 2. Datasets/engine developed using GTA-V.

Dataset/Engine	Task	Description
SAIL-VOS [10]	For studying semantic occlusion segmentation	New dataset containing pixel-level visible and occluded segmentation masks and semantic labels for over 1.8 million objects, with 100 times more annotation than existing datasets.
Crowd Counting [11]	Headcount for crowd scenes	A large, diverse, and comprehensive dataset for crowd counter pre-training was constructed in the GTA-V crowd scenario using the data collector and annotator.
G2D [12]	Ultra-realistic computer-generated images of urban streetscapes for computer vision researchers	Users can interact directly with G2D in-game to manipulate virtual environmental conditions in real time, such as weather, season, time of day, and traffic density, and automatically capture the screen.
Playing for data [13]	For rapid creation of pixel-accurate semantic labelling maps for images extracted from computers	Creates a wrapper that connects the game to the operating system to record, modify, and reproduce rendering commands. Processing rendering resources through hashing allows for the generation of object signatures and the creation of pixel-accurate object labels, eliminating the need to track boundaries.
Sim [14]	Methods for providing datasets primarily for deep training of autonomous driving scenarios	Development of an accelerated deep learning algorithm training method for computer vision and robotics tasks using open-source plugins Script Hook V and Script Hook V.NET to capture synthetic annotated data in the game GTA-V.
JTA [15]	A body part dataset for people tracking in urban scenes	For accurate detection of multiplayer tracking in open-world environments, a virtual dataset was developed using a deep network architecture in order to overcome the difficulties of lack of tracking, and body part occlusion annotation.

When creating synthetic datasets, although virtual environments can be costly to build initially—for example, high-precision 3D modelling often requires high-performance hardware support—they offer significant advantages that make this type of input extremely valuable. Firstly, virtual environments can simulate extreme weather or scenarios that are costly to acquire (e.g., car crashes), which are often difficult to obtain from real-world collections or too costly to acquire at scale. Secondly, virtual environments are highly reusable; after developing a virtual environment once, it can be used multiple times and can be adapted and expanded to meet new needs at a much lower cost than repeated data collection in a real environment. As algorithmic requirements change, virtual environments can quickly generate new scenarios, thus shortening the cycle and cost of updating datasets. This flexibility makes virtual environments an effective tool for data generation that can continuously support the iterative process of algorithm development.

2.2. Eliminating Specular Reflection

So far, only the use of virtual objects in a virtual environment has been demonstrated, all based on existing virtual environments and digital models. However, in reality, it is often not possible to directly obtain the CAD model of the target object. In such cases, object models can be generated through 3D scanning. When using virtual objects in a virtual environment, the reflection on the object's surface is intentionally added to make it look more realistic. However, in 3D scanned objects, reflections are interference factors for the original scan. Therefore, removing reflections is a critical step when using the 3D scanning method.

Some researchers have performed a lot of work on improving image quality and recovering the original information of the image. Zhang Huang et al. [16] proposed an effective specular reflection image enhancement algorithm to address the problem of reduced image quality due to highlights in real-world scenes. Firstly, they obtained a depth map through a colour attenuation prior, then optimised the transmission map with adaptive adjustment factors, and used an L0 gradient minimisation filter to eliminate halo artefacts. Finally, they

employed a non-uniform illumination compensation strategy in the YUV colour space to expand the dynamic range and increase saturation. Zhongqi Wu et al. [17] proposed a novel data-driven approach combining neural networks. They first established a real-image dataset including specular reflection images and diffuse reflection images. Building upon the specular reflection removal network architecture by Wei et al., they introduced Meslouhi et al.'s specular reflection mask input. Weihong Ren et al. [18] proposed a global colour line constraint method based on the dichroic reflection model to separate the specular and diffuse components of an image. They observed that each pixel is distributed along a colour line in the normalised RGB space, and different colour lines represent different diffuse reflection chromaticities and intersect at the light chromaticity. The method uses this global colour line information to efficiently separate the specular and diffuse components of a single image through a pixel-level approach for real-time applications.

There are also researchers who have focused on solving the problem of colour distortion. Traditional colour-based methods often result in significant colour distortions, while local patch-based algorithms fail to incorporate long-range information, potentially leading to artefacts. Hence, Fan Wang et al. [19] proposed a specular reflection removal method based on polarisation imaging through global energy minimisation. Polarisation images provide supplementary information and reduce colour distortion. By minimising a global energy function, they integrate long-range cues and produce accurate and stable results.

Similarly, some researchers have addressed the database shortage problem encountered when deep learning techniques remove specular reflections. Sangho Jo et al. [20] constructed a Specular-Light (S-Light) database for training single-image deep learning models. Additionally, they introduced a multi-scale normalised cross-correlation (MS-NCC) metric to evaluate the learning outcomes, considering the correlation between specular and diffuse reflection components. Zhongqi Wu et al. [21] constructed a large-scale Paired Specular-Diffuse (PSD) image dataset, providing a real-world benchmark dataset for specular highlight removal tasks. They also proposed a new generative adversarial network (GAN) utilising specular reflection information to guide specular highlight removal within a single image. The network employed an attention mechanism to directly model the mapping relationship between diffuse reflection regions and specular highlight regions without explicit illumination estimation. Yakun Chang et al. [22] proposed a ghosting model and synthesised multiple reflection images from a single reflection image using relative intensity to address the issue of insufficient image diversity. They also built a neural network for specular reflection removal, employing a joint training strategy to learn hierarchical separation knowledge from synthesised reflection images to optimise network parameters and simultaneously utilise internal and external losses in the loss function.

Removing specular reflections is crucial in ensuring the accuracy of practical applications. It is the focus of research in many industries.

This is a popular research direction in the field of iris recognition. In 2016, Shahrizan Jamaludin et al. [23] used a pixel attribute method combined with sharpening masking, morphological closing, and flood filling methods to eliminate specular reflection. Firstly, they improved image quality through sharpening masking to enhance contrast and texture. Then, a threshold (set to 40) was applied to the pupil area to detect and determine the position of specular reflection using the pixel attribute method. Next, morphological closing was applied to expand and contract the detected specular reflection. Finally, a flood-filling method was used to eliminate specular reflection. In 2018, they selected and improved the sub-iris technology, reducing the execution time of the iris recognition system and improving recognition accuracy [24]. Two years later, they proposed a fast and accurate specular reflection removal method based on pixel attributes [25].

Specular reflection removal is also a hotspot in the field of endoscopic images. Chao Nie et al. [26] proposed a specular reflection detection and removal method for endoscopic images based on luminance classification. They introduced two new steps: firstly, using an adaptive threshold function that varies with image brightness to detect absolute highlights during highlight detection, and secondly, modifying the priority function of a sample-

based image restoration algorithm during highlight recovery to ensure the rationality and accuracy of restoration. They also adopted a local priority calculation and an adaptive local search strategy to improve algorithm efficiency and reduce error matches. Chi-Sheng Shih et al. [27] treated specular reflection removal as an image-to-image translation task, skipping the specular detection step. The technique was based on dense multi-scale fusion networks (DMFNs) tailored for small-sized and clear images and modified to support large-sized and noisy endoscopic images. Chong Zhang et al. [28] proposed a specular highlight removal method for endoscopic imaging using partial attention networks (PatNets) to reduce the interference of strong light during endoscopic surgery. They segmented endoscopic images based on brightness thresholds and then used partial convolutional networks and attention mechanisms for image restoration.

2.3. Light and Shadow Settings in Virtual Environments

2.3.1. Setting the Shadow

In both Unity and Unreal Engine (UE), there are multiple methods to implement shadow effects, each with its advantages and disadvantages in terms of performance, quality, and applicable scenarios. Tables 3 and 4 summarise the shadow setup methods in Unity [29] and UE [30].

Table 3. Unity shadow setup methods.

Method	Description	Applicable Scenarios
Shadow texture	Creates a flat object with a shadow map and transparent shader for low performance consumption but limited to flat surfaces. Suitable for simple scenes.	Simple scenes, low-quality shadows
Projector projection	Uses the Projector component and Shadow Material to project shadows on different heights with good performance but limited quality. Ideal for medium complexity scenes, balancing performance and effect.	Medium complexity scenes, balancing performance and effect
Spotlight	Utilises a Spotlight object with real-time shadows for high-quality shadows at a high-performance cost. Best for scenes requiring high-quality shadows.	Scenes requiring high-quality shadows
RenderTexture and Projector	Combines RenderTexture with a Projector for pseudo-real-time shadows, balancing quality and performance. Suitable for scenes needing good effects with controlled performance impact.	Scenes needing good effects with controlled performance impact
Shadow map	Captures light and shadow with a camera and RenderTexture, using a shader for high-quality dynamic shadows. Best for complex scenes despite the implementation complexity.	Complex scenes, high-quality dynamic shadows

Table 4. UE shadow setup methods.

Method	Description	Applicable Scenarios
Static shadows	Precompute and bake shadows during the lighting build process; low performance overhead, but shadows remain static and cannot be dynamically updated. Suitable for static scenes.	Static scenes, low performance overhead.
Dynamic shadows	Calculate and update shadows in real time; suitable for scenarios requiring dynamic shadow changes, but may increase performance overhead.	Scenarios requiring dynamic shadow changes.
Hard/soft shadows	Choose between hard or soft shadow effects based on requirements, with hard shadows suitable for strong light environments, while soft shadows offer a more natural appearance.	Choose based on lighting environment requirements.

Table 4. *Cont.*

Method	Description	Applicable Scenarios
Volumetric shadows	Generate realistic light scattering and shadow effects; suitable for complex environments, but with high computational demands.	Complex environments, realistic light scattering and shadow effects.
Distance field shadows	Compute shadows by generating distance field volume data; suitable for large-scale scenes but with high resource requirements.	Large-scale scenes, high resource requirements.
Shadow maps	Utilise rendered depth maps to generate shadows, providing high-quality dynamic shadows; requires careful resolution settings to balance performance and quality.	High-quality dynamic shadows, balancing performance and quality.
Ray-traced shadows	Generate realistic shadow effects using ray tracing technology; demands high-performance hardware.	Realistic shadow effects, high-performance hardware requirements.

2.3.2. Setting up Reflections

Tables 5 and 6 summarise the reflection-setting methods in Unity [29] and UE [30].

Table 5. Unity reflection setup methods.

Method	Description
Reflection probes	Place reflection probes in the scene to capture environmental information, such as surrounding objects' colours, lighting, and reflections. Then, apply these probes to objects requiring reflection effects to simulate surface reflections.
Skybox	Utilise Unity's Skybox feature to simulate the appearance of the sky and reflect it onto object surfaces. By selecting an appropriate Skybox material and applying it to the scene, you can achieve reflection effects by reflecting the environment onto object surfaces.
Custom shader	Write custom shaders to achieve finer control over reflection effects, including specular reflection, refraction, and more. By crafting custom shaders, you can implement various complex reflection effects according to your requirements.

Table 6. UE reflection setup methods.

Method	Description
Reflection capture actors	Capture reflection data of the environment, including nearby objects, lighting, and the sky, by placing these actors in the scene.
Reflection probes	Similar to Unity's reflection probes, these capture environment reflection data and apply it to scene objects, providing accurate reflection information as needed.
Planar reflections	Simulate reflections on flat surfaces like floors, water, or mirrors based on specified plane positions and directions.
Screen space reflections (SSRs)	Compute reflections based on on-screen information, offering performance savings but potentially lower precision compared to other methods.
Material reflections	Create materials with reflection properties to achieve reflection effects, allowing customisation for various characteristics like metallic surfaces or smooth reflections.

2.4. Research Gaps

In today's data-driven research and applications, obtaining and utilising real data faces numerous challenges, including data privacy, data scarcity, and high data annotation costs. As a result, synthetic data, as a supplement to real data, has gradually become a

research hotspot. However, current methods and tools for generating synthetic data have significant research deficiencies in the following areas:

1. Limited customisation: Existing synthetic data generation tools often lack high levels of customisation, making it difficult to precisely tailor them to specific scenarios or specific objects of recognition. This limits the practicality and flexibility of these tools in specific application scenarios.
2. Insufficient data authenticity and diversity: Although synthetic data can partially replace real data, there is still a gap in authenticity and diversity.

This study proposes a virtual environment for generating synthetic data to address the limitations of existing methods. The contributions of this virtual environment are twofold:

1. Enhanced customisation: It provides a high level of customisation, enabling users to tailor the environment precisely to specific scenarios and recognition objects, increasing the applicability of the generated data.
2. Improved data diversity: It supports the simulation of diverse and complex dynamic scenarios, enhancing the realism and variability of the synthetic data to meet different application needs.

3. Methodology

As discussed in the literature review, the method of generating datasets using virtual objects in virtual environments has been validated. However, this approach heavily relies on existing digital models. In many cases, the target model is not readily available, which significantly limits the application scenarios of this method. To address this limitation, this study has chosen to adopt the 3D object scanning approach as a means to obtain target 3D models. Through this method, accurate digital models of real objects can be generated when CAD models are not accessible, thereby ensuring that the advantages of generating datasets in virtual environments can be extended to a wider range of applications. It was also chosen to take into account the fact that some researchers may lack experience and skills in high-quality modelling.

When using virtual objects in virtual environments, reflections on the object's surface are intentionally added to enhance realism. However, in objects generated through 3D scanning, reflections are interference factors introduced during the scanning process. Therefore, removing reflections becomes a critical step when using the 3D scanning method. In the following sections, we introduce the method for specular reflection removal and the use of 3D scanners during the experimental phase.

This study is divided into three main phases, as shown in Figure 1: target modelling, 3D virtual environment construction, and experimental validation. Each phase plays a crucial role in the development and validation of the virtual environment.

a. Target modelling

- Select the 3D target object and create its digital model using a 3D scanner. Chess was chosen as the recognition target for this study.
- After the digital model of the target object has been created, the next step is to optimise its texture to remove highlight reflections. This process ensures that the texture is suitable for use in a 3D environment and does not introduce any artefacts that could affect the accuracy of the object recognition algorithm.

b. Construction of the 3D virtual environment

- After the target model is ready, it is imported into a 3D virtual environment. This environment acts as a virtual world in which various scenarios can be simulated.
- In this environment, various backgrounds and textures are applied, and target objects are placed randomly. This variety is essential to create a dataset with good generalisation.
- This environment implements virtual cameras that capture images from a variety of angles and positions. This virtual camera mimics the behaviour of a real camera and is an important part of generating a realistic dataset.

- The final output is a synthetic image dataset used to train the object recognition algorithm.

c. Experimental validation

- The final output is a synthetic image dataset for training that will be used to train the object recognition algorithm. YOLO v8 and Swin Transformer are chosen as the validation models in this study because YOLO v8 excels in real-time and efficiency and is suitable for fast detection, while Swin Transformer possesses a powerful feature representation and is good at dealing with complex scenes. Leveraging the strengths of both models allows for a more comprehensive evaluation of the dataset's quality and applicability.
- In the evaluation testing phase, the quality of the dataset is assessed by analysing various image data from the model training results, and testing is also carried out on real data images to validate the performance of the algorithms in real applications.

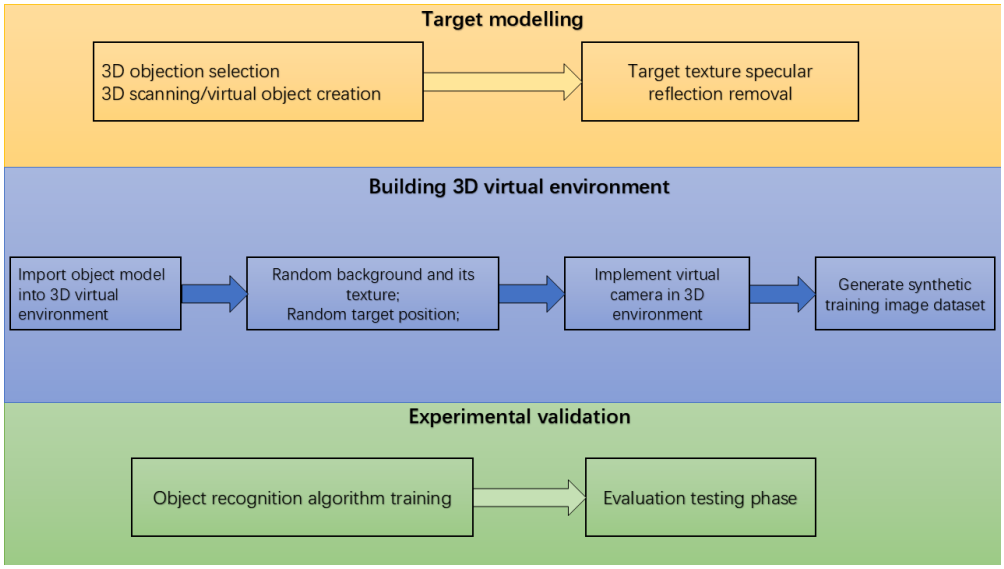

Figure 1. Flowchart of synthetic data generation and their validation.

3.1. Specular Reflection Removal

In this study, 3D scanning is used to create models of target objects. One important issue to note is that 3D virtual objects may carry reflections from the scanning process, particularly specular reflections. These can negatively impact the subsequent use of these objects in virtual environments, as these unrealistic reflections can interfere with the objects' appearance in different scenes and may even mislead recognition algorithms. Therefore, after generating the 3D model, it is necessary to remove these specular reflections to ensure that more realistic simulated reflections can be added in the 3D environment, allowing reflections to naturally change around the object in different images.

The main objective of this study is to establish a virtual space capable of generating datasets. Highlight removal technology aims to restore the texture details of target models and improve their quality in Unity, so a suitable technique must be chosen. In selecting the highlight removal technology, this study primarily considers computational speed and restoration quality. The minimum-value-image method is computationally simple but limited in handling complex lighting conditions; spectral reflection decomposition and Retinex theory provide high-quality restoration but have high computational complexity; the colour space transformation method, while effective, can still exhibit deviations in certain cases. After careful consideration, the luminance separation method was chosen as it not only maintains faster computational speed but also accurately separates and restores the specular and diffuse reflection components in images. Therefore, this study ultimately selects the luminance separation method as the basis for highlight removal.

3.1.1. Dichromatic Reflection Model

At present, the mainstream method of specular reflection removal is based on the dichroic reflection model, which was first proposed by Shafer [31]. The main theoretical content for the incident light on the object's surface is divided into two parts: a part of the incident light passes into the object's interior, with the colourant as a medium, after the reflection leaves the object's surface, i.e., the diffuse reflection of light with the object's own colour information. The other part of the incident light is directly reflected by the surface of the object as specularly reflected light. So according to the dichroic reflection model, the pixel colour can be represented by the sum of its diffuse and specular components.

$$I(x) = D(x) + S(x), \tag{1}$$

where $I(x)$ denotes $[I_r(x), I_g(x), I_b(x)]$ for each pixel, $D(x)$ denotes the diffuse component, and $S(x)$ denotes the specular component, also called illumination chromaticity. The formula can also be rewritten as

$$I(x) = w_d(x)\Lambda(x) + w_s(x)\Gamma(x), \tag{2}$$

where w_d and w_s are the weights associated with diffuse and specular reflections on the surface geometry: $\Lambda = [\Lambda_r(x), \Lambda_g(x), \Lambda_b(x)]^T$, $\Gamma = [\Gamma_r(x), \Gamma_g(x), \Gamma_b(x)]^T$. Typically, it is assumed that the illumination chromaticity of a given image is uniform, leading most scholars to initialise the illumination chromaticity $\Gamma(x)$ as $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$.

3.1.2. Separation of Diffuse and Specular Reflections

Specular highlights are areas of glare caused by light reflecting directly onto the camera lens and are usually found in the highlighted portions of an image, making these areas significantly brighter than the surrounding areas. It primarily affects the brightness of the image, rather than the chromaticity, and therefore increases the brightness of certain pixels, making their original colour information relatively diminished. Specular reflection usually behaves as a near-white light because it reflects the colour of the light source, which tends to be white.

Mainstream highlight removal techniques, whether it is the minimum-value-image method, chromaticity space clustering method, colour space transformation, spectral reflection decomposition, or Retinex theory, all require normalising the colour vectors first:

$$I_{\text{linear}}(x) = \frac{I(x)}{I_r(x) + I_g(x) + I_b(x)}. \tag{3}$$

Next, pixel clustering is performed [18]. First, the colour vector I_{linear} is transformed into the spherical coordinate system, resulting in three parameters: longitude $\theta(x)$, latitude $\phi(x)$, and the distance to the illumination chromaticity $r(x)$. The longitude and latitude represent the chromaticity of the pixel, while the distance reflects the specular reflection component of the pixel. The chromaticity distance $r(x)$ between two pixels is defined as the sum of the absolute differences of their longitudes and latitudes. If the chromaticity distance between two pixels is less than a preset threshold T, they are assigned to the same cluster. The specific clustering steps are as follows: first, randomly select initial cluster centres; then, assign each pixel to the nearest cluster centre based on chromaticity distance; next, calculate the new centre of each cluster as the average position of all pixels assigned to that cluster; repeat the assignment and updating steps until the cluster centres no longer change significantly. Once clustering is complete, each cluster corresponds to a colour line, with their intersections accurately estimating the illumination chromaticity.

3.1.3. Separation of Diffuse and Specular Reflections

The intersection of all the chromaticity lines is also the value of the illumination chromaticity $\Gamma(x)$ according to Equation (4) in [18]:

$$D(x) = \alpha(x)\Lambda(x) = I_{\text{linear}}(x) - (1 - \alpha(x))\Gamma(x). \quad (4)$$

$\alpha(x)$ represents the mixing degree between the diffuse chromaticity and the illumination chromaticity for a pixel. It is calculated by dividing the distance from the pixel to the illumination chromaticity $r(x)$ by the maximum distance $r_{\max}(x)$ among all pixels in the cluster. To reduce noise, the final $\alpha(x)$ is processed with a 5×5 median filter. Additionally, to avoid over-separating specular reflections, the maximum distance is replaced by the median distance $r_{\text{median}}(x)$ of all pixels in the cluster during the actual calculation. Finally, D(x) is transformed into RGB chromaticity space. The key principles of highlight reflection removal have been introduced and are presented in Algorithm 1.

Algorithm 1 Specular Reflection Separation Algorithm

Input: Highlighted image $I(x)$
Output: Diffuse reflection $D(x)$, Specular reflection $S(x)$
// Step 1: Linearize color vector
for each pixel x in $I(x)$ **do**
    $I_{\text{linear}}(x) \leftarrow \frac{I(x)}{I_r(x)+I_g(x)+I_b(x)}$
end for
// Step 2: Convert to spherical coordinates
for each pixel x **do**
    Convert $I_{\text{linear}}(x)$ to spherical coordinates to obtain $\theta(x)$, $\phi(x)$, $r(x)$
end for
// Step 3: Pixel clustering
Randomly initialize cluster centres in spherical coordinates
repeat
    for each pixel x **do**
        Assign x to the nearest cluster centre based on chromaticity distance:
        $|\theta(x) - \theta(\text{center})| + |\phi(x) - \phi(\text{center})|$
    end for
    Update each cluster centre as the mean position of all pixels in the cluster
until cluster centers converge
// Step 4: Fit the colour line and estimate illumination chromaticity
for each cluster **do**
    Fit color line and calculate illumination chromaticity $\Gamma(x)$
end for
// Step 5: Calculate distance to illumination chromaticity
for each pixel x **do**
    $r(x) \leftarrow \|I_{\text{linear}}(x) - \Gamma(x)\|$
end for
// Step 6: Calculate maximum distance $r_{\max}(x)$ for each cluster
for each cluster **do**
    $r_{\max}(x) \leftarrow \max(r(x))$ for all pixels in the cluster
end for
// Step 7: Compute diffuse coefficient $\alpha(x)$
for each pixel x **do**
    $\alpha(x) \leftarrow \frac{r(x)}{r_{\max}(x)}$
end for
Apply a 5×5 median filter to $\alpha(x)$
// Step 8: Compute diffuse component $D(x)$
for each pixel x **do**
    $D(x) \leftarrow \alpha(x)\Lambda(x) = I_{\text{linear}}(x) - (1 - \alpha(x))\Gamma(x)$
end for
Convert $D(x)$ back to RGB color space
// Step 9: Compute specular component $S(x)$
for each pixel x **do**
    $S(x) \leftarrow I(x) - D(x)$
end for
return $D(x)$, $S(x)$

3.1.4. GPU Acceleration

Souza et al. [32] used a GPU to greatly improve the computational efficiency of their algorithm when performing highlighting, so this method is also used in this study to reduce the algorithm execution time. Because the highlight removal algorithm utilised in this study involves a large number of independent computations, it is suitable to be allocated to multiple threads for simultaneous processing. Examples include the conversion of the image from RGB space to the spherical coordinate space process, distance calculation, and diffuse reflection coefficient estimation. For finding the minimum distance, optimisation can be achieved using parallel prefixes. The GPU used in this study is an NVIDIA GeForce RTX 3060 Laptop with 3840 CUDA cores and supports Direct3D feature level 12.1.

3.2. Establishment of the Virtual Environment

The virtual environment is mainly based on Unity and is developed based on the Unity Perception package provided by Unity officially.

The Unity Perception package is a tool for generating synthetic data, primarily for training computer vision models. It simplifies the process of creating and managing 3D scenes, significantly reducing the time and cost of data collection. An overview of the establishment of the virtual environment for this project is shown in Figure 2.

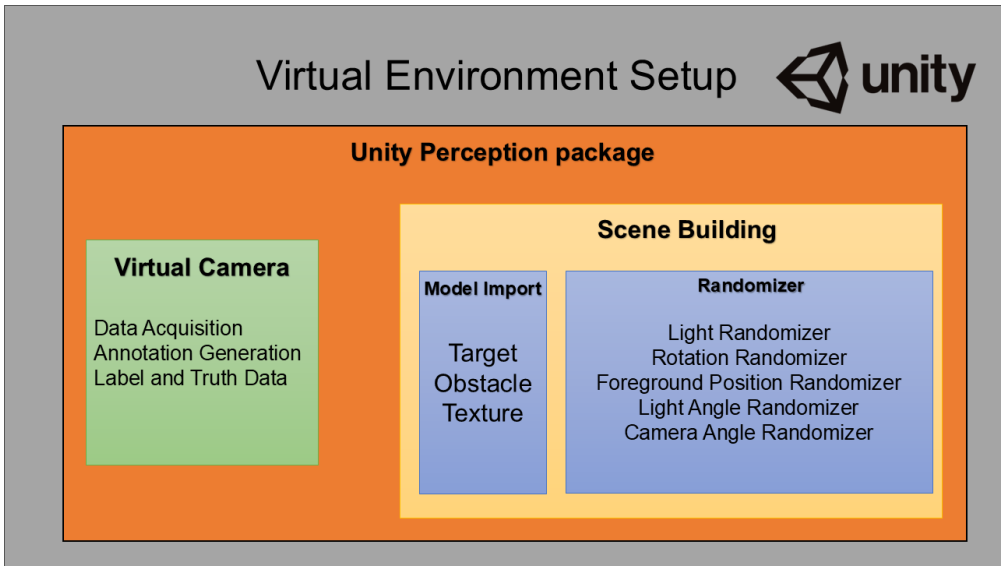

Figure 2. Overview of virtual scene creation.

3.2.1. Virtual Camera

The Unity virtual camera is used to capture and generate image data with annotations. These data are primarily used to train and evaluate computer vision models. Here are the main features and functionality of the Unity virtual camera [29]:

1. Data acquisition and synthesis: Various types of data can be generated in the virtual environment, including RGB images, depth images, semantic segmentation images, instance segmentation images, etc.
2. Annotation generation: Annotations corresponding to the image data are automatically generated, including the category, position, bounding box, and occlusion information of the objects. These annotations can be used to train machine learning models.
3. Label and truth data: Provides accurate labels and truth data, which may be difficult to obtain in the real world. For example, it is possible to accurately label the poses and key points of objects.
4. Data export: The generated images and annotated data can be exported to solo format for easy integration with machine learning frameworks (e.g., TensorFlow, PyTorch).

Also, this project takes advantage of the virtual camera's ability to output depth images, which helps in the training of the monocular camera recognition distance model.

3.2.2. Randomiser

The randomiser is used to generate diverse synthetic datasets by randomising the attributes of scene elements to increase the diversity of the dataset. The following are the main functions of the randomiser:

1. Object attribute randomisation: Randomises attributes such as colour, texture, position, rotation, and scaling of objects.
2. Scene element randomisation: Randomises lighting conditions, background, camera parameters, etc., in the scene.
3. Custom randomisation: Users can write custom randomisers to flexibly control the randomisation logic of any scene element.
4. Randomisation frequency control: Set the frequency of randomisation, e.g., every frame, every N frames, or based on specific event triggers.

To increase the diversity of the dataset in this project, the following randomisers were mainly utilised:

1. Light randomiser: Randomises light source properties in the scene, such as light source type, intensity, colour, and position.
2. Rotation randomiser: Randomises the rotation angle of an object so that it is rendered at different angles in the scene.
3. Foreground position randomiser: Randomises the position of target recognition objects to increase the diversity of object positions in the dataset.
4. Light angle randomiser: Randomises the angle of the light source to simulate different lighting conditions and increase the diversity of the dataset.
5. Camera angle randomiser: We also develop a camera angle randomiser based on a custom tutorial to simulate multi-angle camera shooting situations.

3.2.3. Background Selection Criteria

In constructing the virtual environment, background selection criteria are key to ensuring dataset quality and enhancing model performance. This study adopts the principle of "primarily matching real-world contexts, with supplementary irrelevant objects or colours". This approach not only realistically replicates typical scenarios for target objects but also introduces moderate complexity to increase data diversity, thereby enhancing model robustness and ensuring more stable performance in complex scenes.

1. Primarily matching real-world contexts: The main basis for background selection is the typical real-world contexts in which the target object is commonly found, ensuring the object appears in a realistic and plausible setting within the virtual environment. This helps the model accurately learn the relationship between the target object and its usual environment, enhancing its performance in real-world applications.
2. Supplementing with irrelevant objects or colours: While maintaining background realism and relevance, we moderately introduce irrelevant objects or colours to simulate real-world complexity. Such distractions increase scene diversity during model training, allowing for better adaptation to varied environments and enhancing robustness against interference.

3.2.4. Processing of Datasets

The workflow for generating a dataset using the Unity Perception package is shown in Figure 3: first, import the 3D model, including target objects, obstacles and textures. This step ensures that all required assets have been loaded into the Unity project.

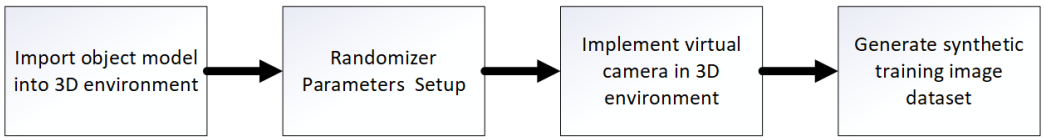

Figure 3. Dataset generation process.

Then, the randomisation parameters of the scene are configured. This step is very important because it increases the diversity and robustness of the dataset. The randomisation parameters include light source randomisation, rotation randomisation, foreground position randomisation, light angle randomisation, and camera angle randomisation. These randomisation settings allow multiple scenes to be generated under different conditions, thus improving the effectiveness of model training.

Next, the virtual camera is set up, which is key to data acquisition. The camera configuration needs to include data acquisition, annotation generation, and recording of labels and real data. These configurations ensure that the camera captures high-quality images and relevant metadata. It also includes internal and external camera parameter settings, either default settings or physical camera options, so that the camera vision is closer to a real camera.

Finally, the scene is run and the data are exported. This step, implemented through the Unity Perception package, executes the predefined scene and randomisation settings and captures the generated data. Once the data are captured, they are exported for further analysis and use.

The default output dataset format of the Unity Perception package is solo because we chose YOLO v8 and Swin Transformer as the recognition model, so we chose the solo2yolo and solo2coco tool in pysolotools to convert the solo dataset to YOLO and COCO formats. Then, the dataset is enhanced with dataset enhancement, including flip, rotate, shift, crop, deform, scale, noise, blur, erase, and other types of operations to further enrich the dataset and improve its robustness.

3.3. Monocular Ranging

The main objective of ranging in this subsection is to transform object recognition into 3D positioning of the object's location and orientation in order to verify whether the dataset created through the virtual environment conforms to real-world physical laws, as well as the usability of the synthetic data in ranging algorithms. This is not aimed at improving ranging accuracy; therefore, only the basic principles of monocular ranging are utilised.

4. Experiment and Results

This section consists of two main parts. The first part covers the 3D scanning hardware and software used in the experiment and the configuration of the virtual environment. The second part presents the image preprocessing results and analyses the performance of different object recognition models on the dataset, along with their test results in real-world scenarios.

4.1. Experimental Preparation

4.1.1. Target Acquisition

The experimental phase mainly uses 3D scanning equipment to capture the target data so that its model can be imported into Unity's virtual environment for dataset generation. The scanning hardware facilities for this project are shown in Figure 4.

Device 1 is the Artec space spider (Central Scanning Limited, Bromsgrove, Worcestershire, B61 0BX, United Kingdom), a high-precision 3D scanner developed by Artec 3D for use in areas requiring high resolution and fine detail, such as reverse engineering, product design, manufacturing, quality control, and medical applications. Featuring a scanning accuracy of up to 0.05 mm and a dot pitch of 0.1 mm, it is capable of fast scanning at 1 million dots per second for complex geometries. It is designed to be intuitive and easy to use for a wide range of users and features a built-in temperature stabilisation system to

ensure stable operation at different temperatures. The unit is compatible with a wide range of 3D modelling and CAD software for easy data processing and application [33]. Device 2 is a Dell Precision 7710.

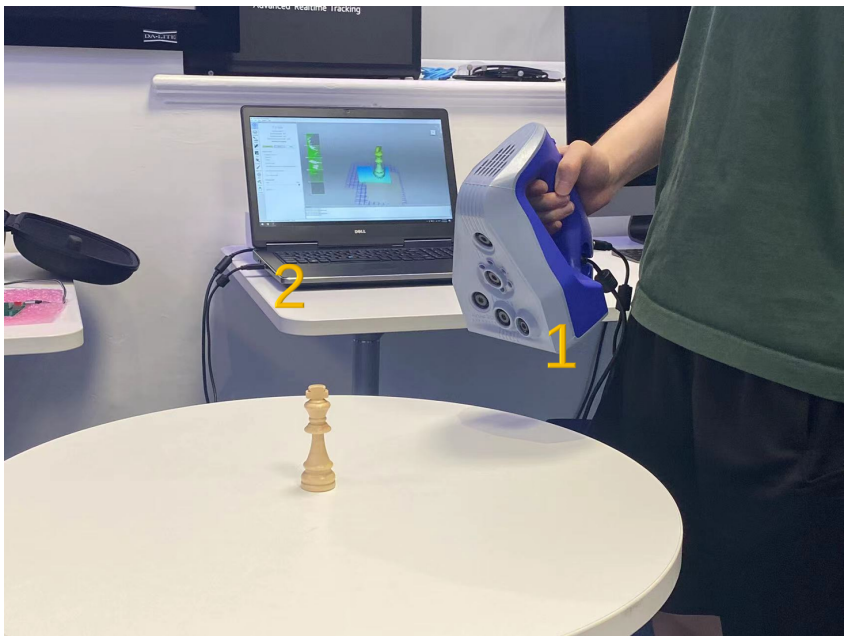

Figure 4. The 3D scanning experiment: 1 refers to device 1 and 2 refers to device 2, respectively.

The software facility is Artec 14 Studio Professional, a powerful 3D scanning and data processing software, also developed by Artec 3D, designed to provide high-precision 3D scanning and post-processing capabilities. It is able to easily convert scanned data into high-quality 3D models with automatic alignment, clean-up, and filling. It handles high-resolution data, captures minute details and complex geometries, and is compatible with a wide range of 3D scanners and mainstream CAD software.

Figure 4 shows a scene from a 3D scanning experiment. An Artec spider 3D scanner was used in the experiment and scanned a chess piece fixed to a stand on the table. The laptop screen in the centre shows the 3D model generated in real-time during the scanning process, where the scanner transfers the surface data of the object to the computer and performs a 3D reconstruction.

4.1.2. Virtual Environment Creation

Currently, the Unity Perception package [34] is still in the experimental stage, mainly adapted to Unity version 2021.3.37f1c1.

The objects obtained from the 3D scanner were imported into Unity, including models in OBJ format and textures in PNG or JPG format. The primary targets for this experiment were the pawn, rook, knight, and bishop. After being acquired by the 3D scanner, these objects were first optimised and processed in Artec Studio 14 Professional software to ensure that they would display and perform optimally in Unity. The main steps included tools for surface smoothing and trimming excess data. Surface smoothing improves model quality by removing irregularities in the mesh using tools such as smoothing brushes. Trimming is the process of selecting and removing excess areas from the scanning process to ensure a clean and accurate final model.

The first thing we did was to turn our imported models into prefabs; the Unity Perception package provides the ability to quickly create prefabs from multiple models. Simply select all models and choose 'Assets → Perception → Create Prefabs from Selected Models' from the top menu bar and the newly created prefabs are placed in the same folder as their corresponding models. The point of prefabs is to make it easy to manage and organise objects, improve reusability, support dynamic generation of objects, and

ensure consistency across multiple instances. This approach simplifies object management, improves development efficiency, and ensures consistent updates to object properties.

Next, the scene was configured in Unity and the camera and light sources adjusted. The camera position was set to (0, 0, 27) and the physical camera option was ticked, with the aim of making the final image close to the quality of an actual camera image. A virtual camera script was hooked up to the camera; it was used to modify aspects of composite frame generation and annotation. The light source was adjusted, with the main purpose of adjusting the quality of the generated shadows.

At the same time, the tagger in the camera was primarily set up to automate data labelling. The Perception package provides the following commonly used taggers, and in this study, we mainly use the 2D bounding boxes and instance segmentation labellers. Also, the ID Label Config provided by the Perception package was created to hold the names identified by the labellers. The labelling script for the imported model prefab was hooked up, the model was named and the name was synced to the ID Label Config file.

In order to perform a randomisation simulation, you need to create a new empty object in the scene called simulation scenario and add the fixed length scenario component. Then, add the randomiser. The main randomisers used in this study were the light randomiser, the rotation randomiser, foreground position randomiser, light angle randomiser and camera angle randomiser. Then, the relevant parameters of the randomiser were arranged, and the rotation angle of the target was set between -60 degrees and 60 degrees for the X-axis, and between -30 degrees and 30 degrees for the Y-axis and Z-axis after several adjustments. In addition, some open models and textures on the web were selected as backgrounds and obstacles to enrich the diversity of the dataset and effectively improve the robustness and anti-interference ability of the dataset.

4.2. Results

4.2.1. Specular Highlight Removal

In this section, the primary goal is to restore the original texture information on the surface of the object by specular reflection removal technology, which makes the scanned model more realistic in the virtual space. Two sets of comparison images are shown below to illustrate the changes in the image before and after highlight removal.

Figures 5 and 6 show the effect of specular reflection removal technology on different images. Each set of images is divided into four columns: a: the original image, which contains highlights; b: the processed image, where highlights are removed; c: the heat map of the original image, which shows the heat distribution in the highlighted areas; and d: the heat map of the processed image, which shows the change in the heat distribution after the highlights are removed. The two images show the surface of the object under different angles and lighting conditions, and it can be seen that the details and texture information on the surface of the object have been significantly restored after the highlight removal. Especially from the comparison of (a) and (b), it can be noticed that the processed image reduces the influence of the highlighted area, which reveals the texture details originally covered by the highlights. This is further verified by thermal maps (c) and (d), where the processed thermal maps show a more uniform temperature distribution, reducing the abnormally high temperatures in the specular reflection area.

Table 7 presents image quality metrics for the highlight removal technique, including VIF, SSIM, MSE, and PSNR. VIF measures visual information loss, SSIM evaluates image similarity based on structure, contrast, and luminance, MSE calculates the average squared error between images, and PSNR assesses the ratio of signal strength to noise. These metrics are commonly used for image quality assessment.

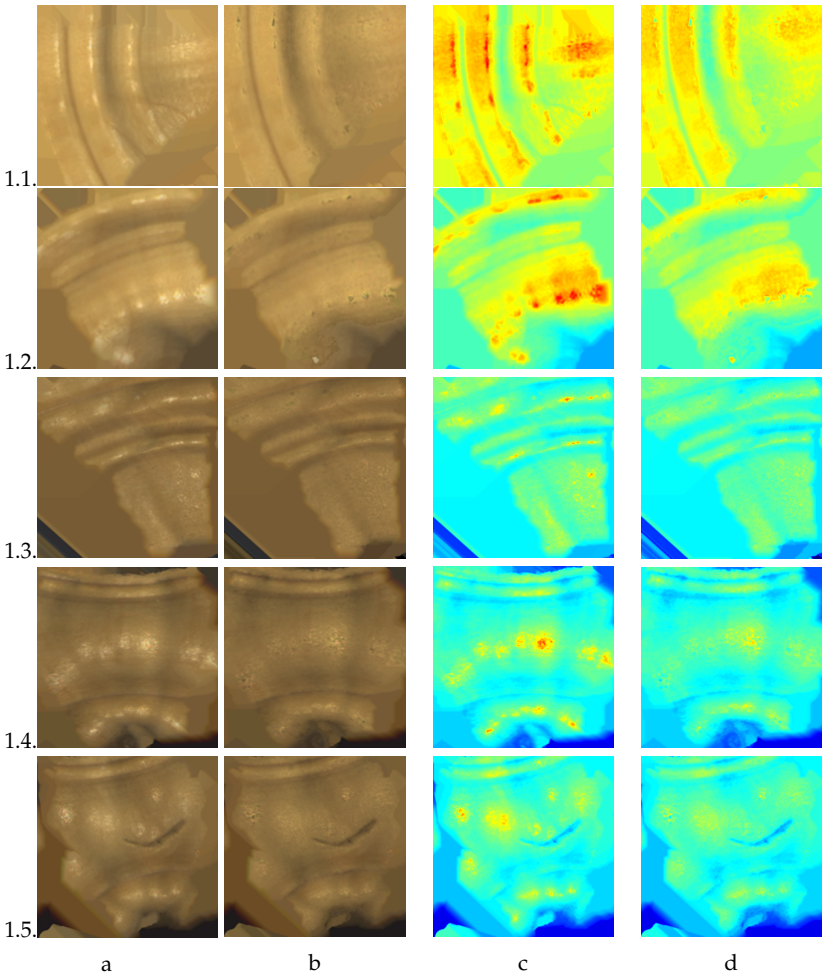

Figure 5. Highlight removal comparison part 1, where different rows indicate scanned data for different objects. (**a**) Original images; (**b**) processed image; (**c**) original images (heat map); (**d**) processed image (heat map).

Table 7. Image quality metrics.

Image Pair	VIF	SSIM	MSE	PSNR
1.1	0.0255	0.9950	98.3681	28.2023
1.2	0.0617	0.9960	78.5781	29.1778
1.3	0.5122	0.9932	51.7563	30.9912
1.4	0.8430	0.9886	99.1844	28.1664
1.5	0.3229	0.9951	34.6164	32.7380
2.1	0.7991	0.9982	14.4308	36.5379
2.2	0.3646	0.9983	16.6276	35.9225
2.3	0.8441	0.9979	16.4571	35.9673
2.4	0.3445	0.9945	76.5607	29.2907
2.5	0.4334	0.9953	45.8617	31.5163

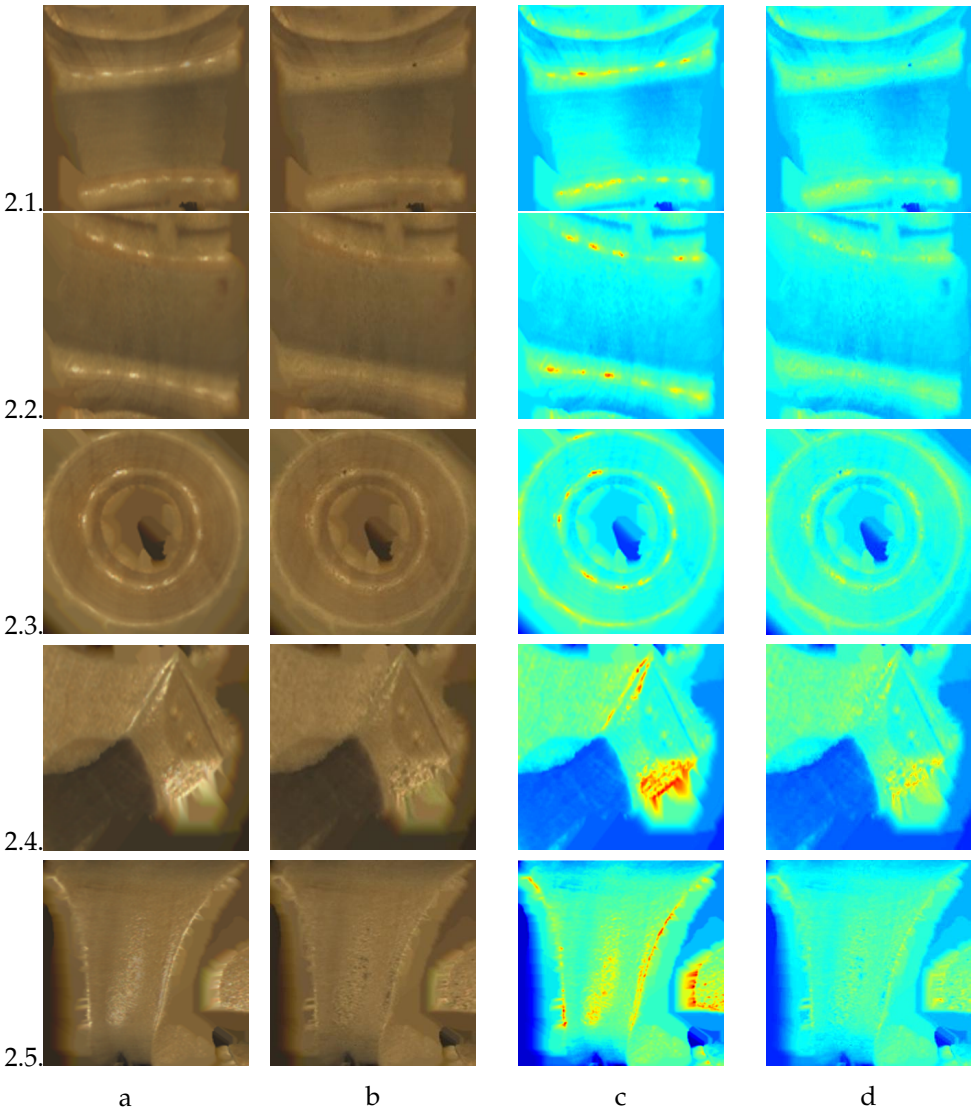

Figure 6. Highlight removal comparison part 2, where different rows indicate scanned data for different objects. (**a**) Original images, (**b**) processed image, (**c**) original images (heat map), (**d**) processed image (heat map).

4.2.2. 3D Model Scanning

Figure 7 illustrates the scanned models of the knight (a, b) and bishop (c, d) (first row) compared to the actual models (second row). It is clear from the figure that the 3D scanned models accurately capture the details and shapes of the actual models, including the complex textures and contours. The object scans are very similar in shape and appearance, and with the removal of specular reflections, they are expected to be sufficient for training an object recognition model, allowing the research to progress to the next stage.

4.2.3. Synthetic Datasets

Unity is used to simulate the process of falling chess pieces, and then the camera is used to capture the images of the process, which improves the diversity and richness of the generated dataset. Simultaneously, the software can perform several iterations, in order to simulate the falling and collision of the chess pieces. This allows the generation of the object in different locations and at different orientations with respect to the virtual camera, with varying lighting conditions. As shown in Figure 8, a partial display of the dataset is displayed. In Figure 9, the chess pieces have been isolated using semantic segmentation, which automates

the process of identifying and separating objects in the image, rather than manually selecting regions. Figure 10 illustrates the results of capturing images during one continuous collision.



Figure 7. Comparison of scanned models and actual models. (**a**) White knight, (**b**) Black knight, (**c**) White bishop, (**d**) Black bishop.



Figure 8. Presentation of synthetic dataset.

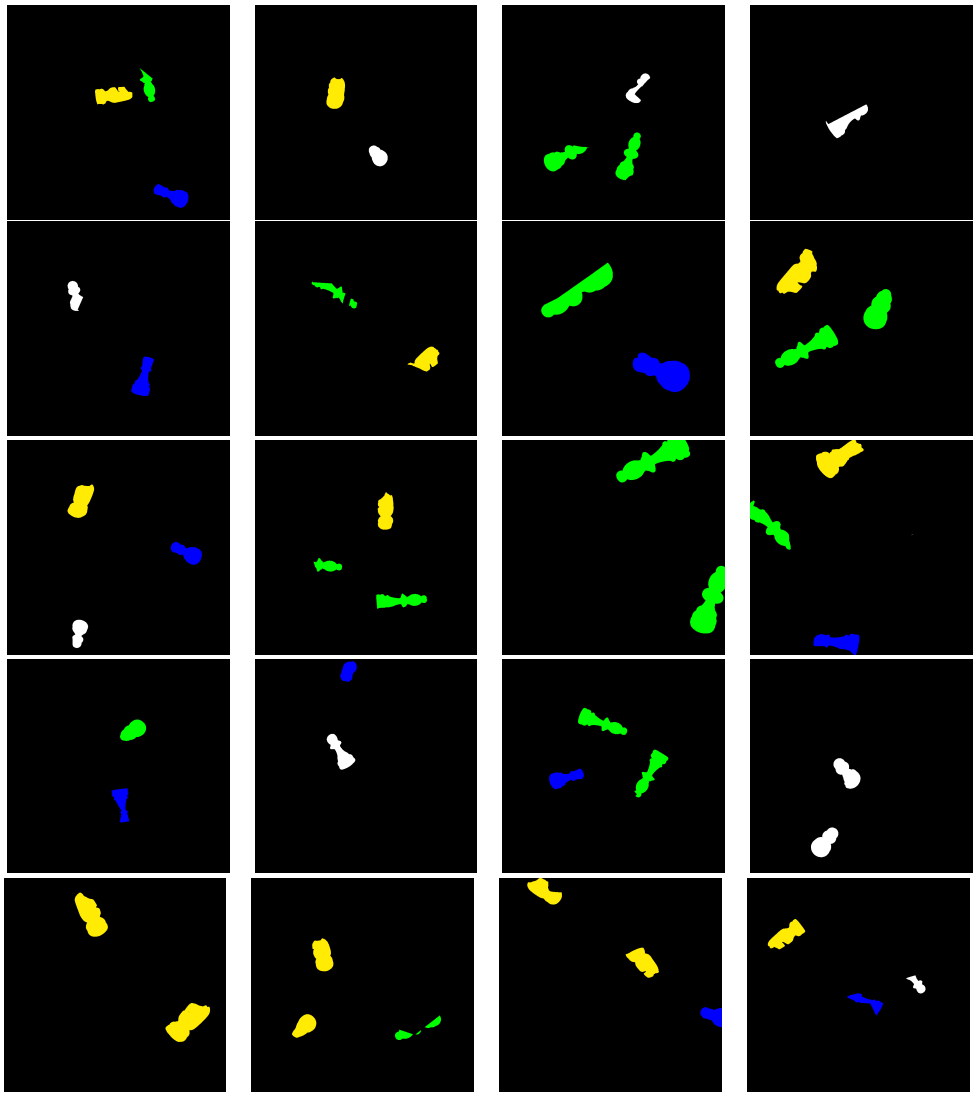

Figure 9. Demonstration of semantic segmentation graph. Yellow represents the Knight, white represents the Pawn, green represents the Bishop, and blue represents the Rook.

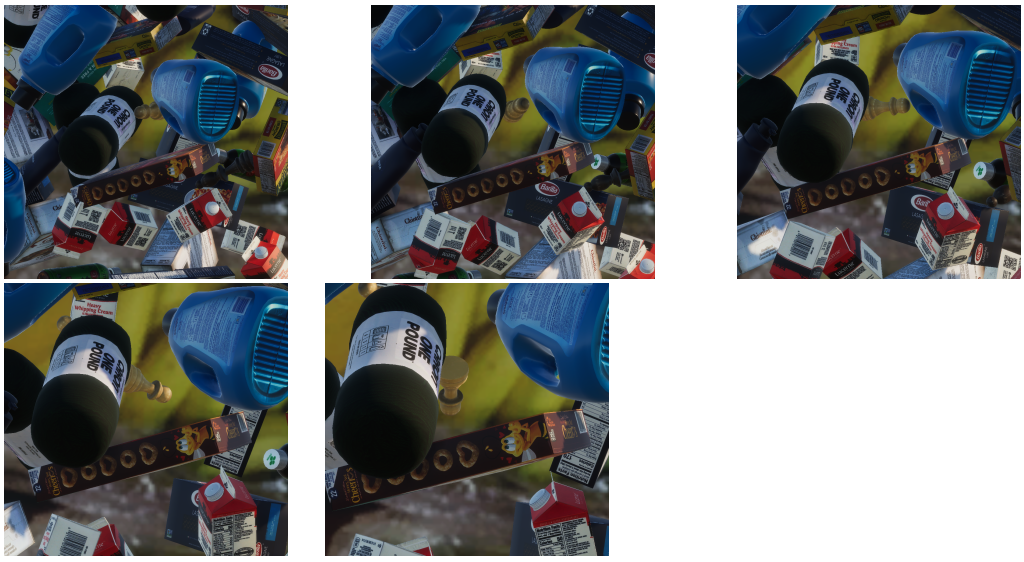

Figure 10. Capture images during continuous collisions.

Many current studies are based on depth imagery, but in this study, this feature was used primarily to demonstrate the diversity and functionality of the environment rather than as a primary research methodology. Therefore, the purpose of generating these depth images was for demonstration purposes only and was not further applied in the subsequent research. Figure 11 shows the generated depth images.

By observing the images, the following issues should be noted:

1. Image consistency issue: The depth information across different parts of the image is inconsistent. In this study, the depth images are merely an add-on to the synthetic data generation and do not represent a unified depth scene.
2. Background clutter issue: The main reason for this issue is that during the generation of synthetic data, the images were primarily captured during the process of model objects falling and colliding. The depth images, being an add-on, also reflect this process.

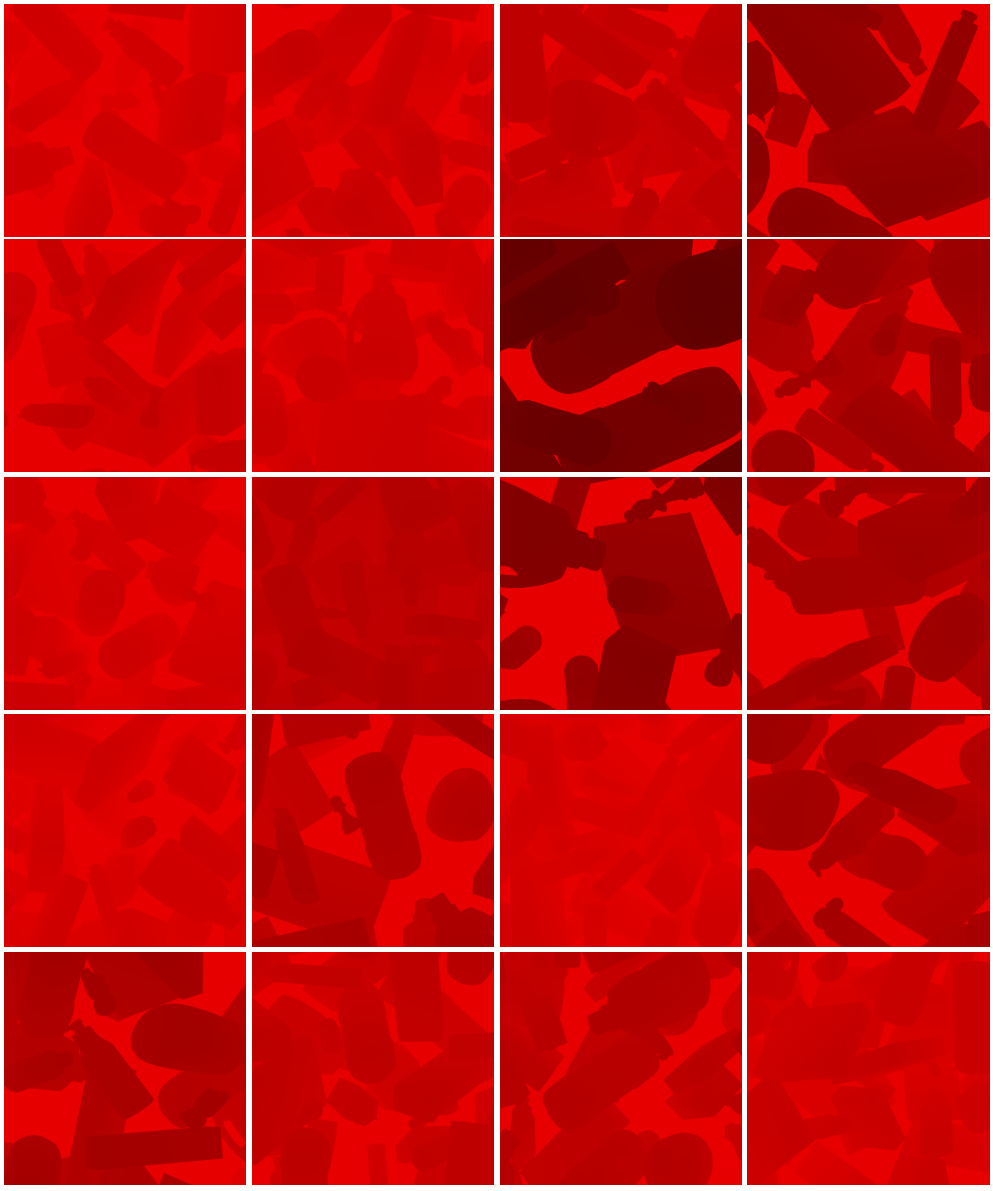

Figure 11. Depth images.

4.3. Validation Dataset

4.3.1. YOLO Validation Dataset

In Figure 12, the F1 score, precision, and recall of our classification model, with the confidence level, as well as the precision–recall curve, are demonstrated. As can be seen from the figure, different categories (rook, bishop, pawn, knight) perform well on all metrics, and all categories reach an F1 score of 1.0 at a confidence level of 0.516 and a precision of 1.0 at 0.947, which indicates that the model has high accuracy and reliability at high confidence levels. Especially, the precision–recall curve shows that the mAP@0.5 for all classes reaches 0.995.

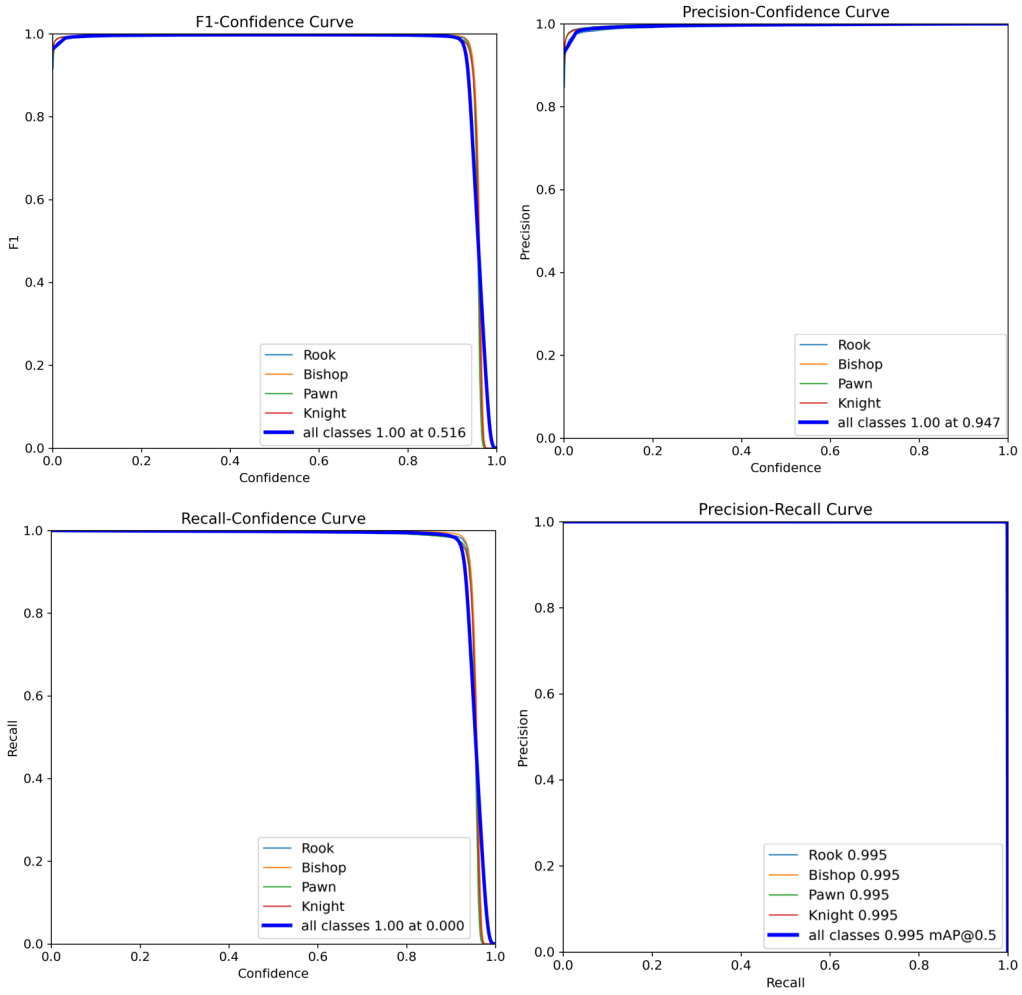

Figure 12. Curves of F1 score, precision, and recall versus confidence for different categories.

As shown in Figure 13, the confusion matrix demonstrates the classification results of the model on the test set. These four charts display different characteristics of the dataset: the bar chart in the top left shows the number of instances for each category (rook, bishop, pawn, knight); the heat map in the bottom left illustrates the x and y position distribution of the bounding boxes within the images; the heat map in the bottom right presents the width and height distribution of the bounding boxes; and the small chart in the top right displays the overlap of bounding boxes, reflecting the layout of different categories within the images. As can be seen from the figure, the classification between different categories is better and most of the instances are correctly classified. The bishop category was the most accurately classified, with fewer instances misclassified. Overall, the model has a high classification accuracy across categories, further illustrating the usability and effectiveness of the synthetic dataset. Figure 14 shows the distribution of instances of different categories

in the synthetic dataset and the corresponding heat map. The histogram shows that the bishop category has the most instances at about 40,000, while rook, pawn, and knight have a relatively balanced number of instances, all at about 25,000. Additionally, it is important to clarify that these numbers represent the frequency of instances within the segmented (labelled) training dataset. The heat map below shows the distribution of the x and y positions of the objects in the dataset as well as the width and height distribution. It can be seen that the x and y distribution of the objects is relatively uniform, while the width–height distribution is concentrated in a specific range, indicating that the dataset has good uniformity and representativeness in the generation process. The number of bishop images is higher because their complex features require more examples to improve the final accuracy.

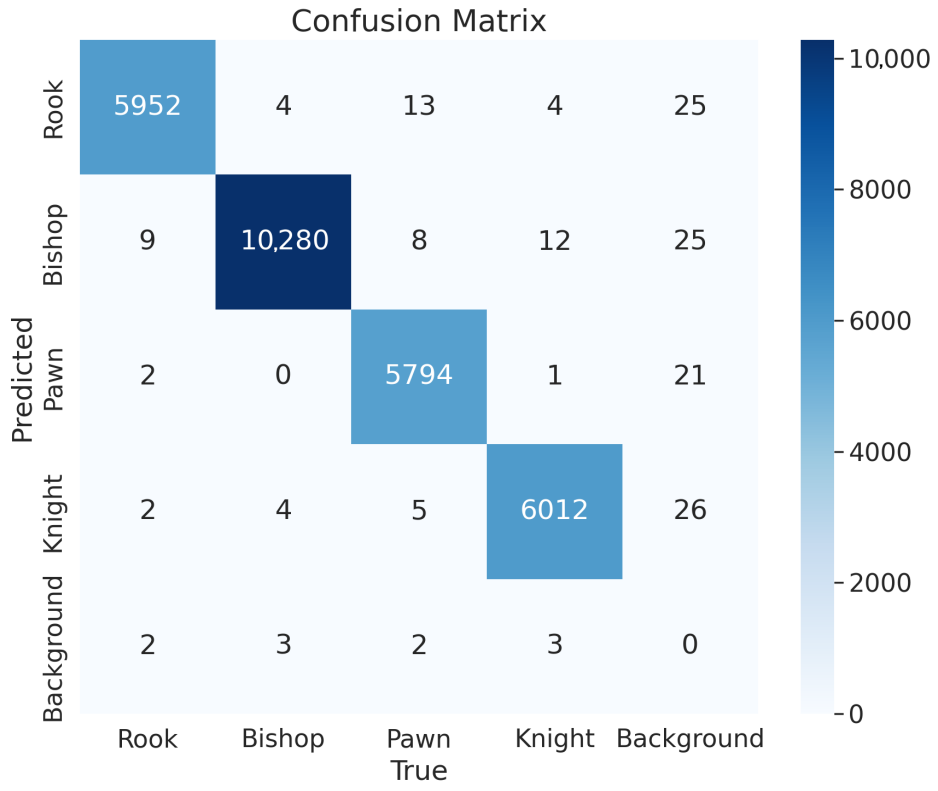

Figure 13. Confusion matrix.

Figure 15 illustrates the variation in losses and metrics (precision, recall, and mAP) during the training and validation process. It can be observed that with the increase in training iterations, both training and validation losses gradually decrease, while model precision and recall significantly improve and stabilize. Notably, mAP@0.5 and mAP@0.5 to 0.95 on the validation set reach nearly 1.0, indicating the excellent performance of the model across different IoU (intersection over union) thresholds. These results further demonstrate the effectiveness of using the Unity-synthesised dataset for training deep learning models.

Table 8 shows the detection results of the monocular ranging system, including the target object, virtual environment coordinates (the world coordinates of the target object set in the virtual environment), and algorithm-inferred coordinates (the world coordinates of the target object inferred from the synthetic images using the ranging algorithm). The default coordinates of the virtual camera in the virtual environment are $(0, 0, -27)$.

Table 9 illustrates the orientation vectors of the individual targets with respect to the camera.

Figure 16 demonstrates the results of using the YOLO algorithm to recognise pictures of real pieces. Although the model is trained with purely synthetic data, the recognition

results are clearly labelled with the positions and confidence levels of different pieces (i.e., bishop, knight, rook, and pawn), which indicates that the model still performs accurately in a real-world environment.

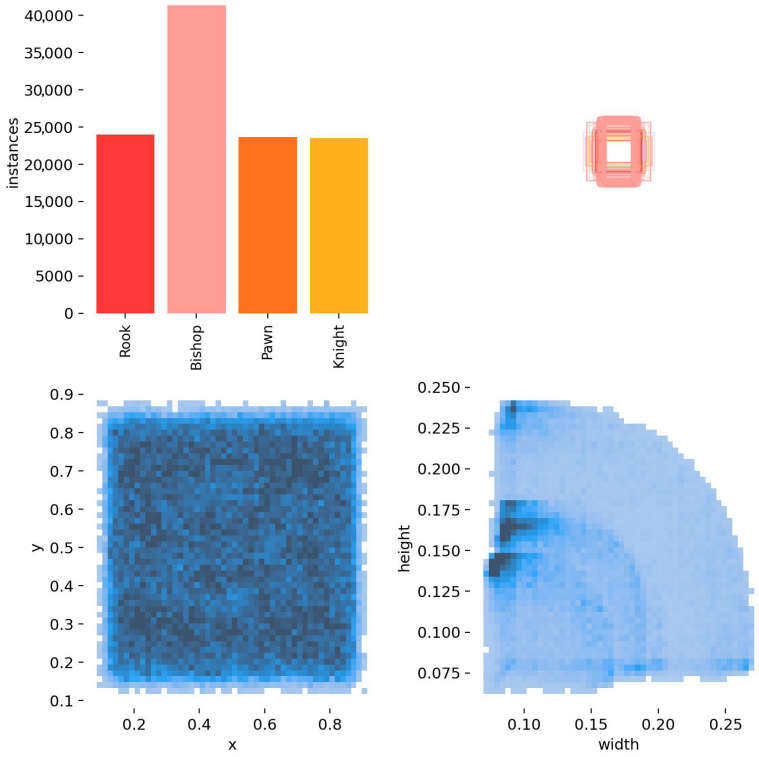

Figure 14. Heat map of data distribution and target box locations.

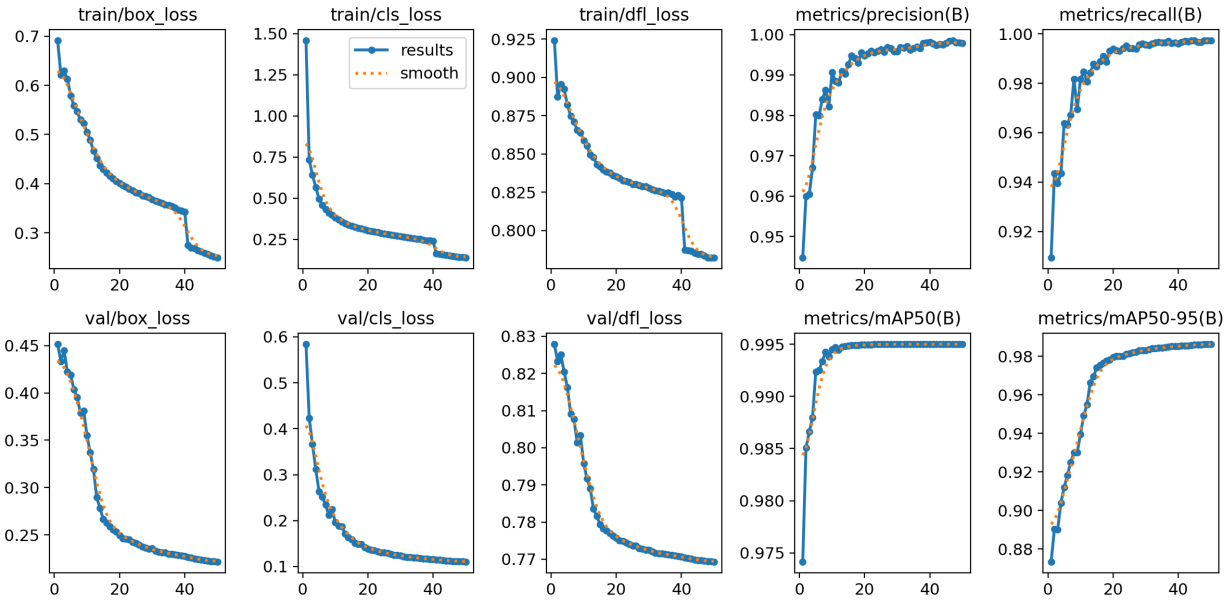

Figure 15. Losses during training and validation and changes in evaluation metrics.

Table 8. Position detection data.

No.	Target	Actual World Coordinates	Algorithmic World Coordinates
1	Bishop	(0, 0, −1)	(0, −0.14, −0.8)
2	Bishop	(0, 1.5,−1)	(0, 1.42, −1.08)
3	Bishop	(0, −1.5, −1)	(0, −1.71, −0.83)
4	Bishop	(1, 0, −1)	(1.05, −0.14, −0.77)
5	Bishop	(−1, 0, −1)	(−1.05, −0.14, −0.78)
6	Knight	(0, 0, −1)	(0.05, −0.18, −0.95)
7	Knight	(0, 1.5, −1)	(0.05, 1.38, −0.80)
8	Knight	(0, −1.5, −1)	(0.05, −1.75, −0.8)
9	Knight	(1.5, 0, −1)	(1.61, −0.18, −0.91)
10	Knight	(−1.5, 0, −1)	(−1.51, −0.18, −1.03)

Table 9. Corresponding orientation.

No.	Target	Relative Direction (Camera)
1	Bishop	(0.0, −0.0026, 0.999997)
2	Bishop	(0.0, 0.0268, 0.99964)
3	Bishop	(0.0, −0.0321, 0.99948)
4	Bishop	(0.0197, −0.0026, 0.99980)
5	Bishop	(−0.0197, −0.0026, 0.99980)
6	Knight	(−0.0009, −0.0033, 0.999993)
7	Knight	(−0.0009, 0.0259, 0.99966)
8	Knight	(0.0009, −0.0328, 0.99945)
9	Knight	(0.0303, −0.0034, 0.99953)
10	Knight	(−0.0285,−0.0034, 0.99958)

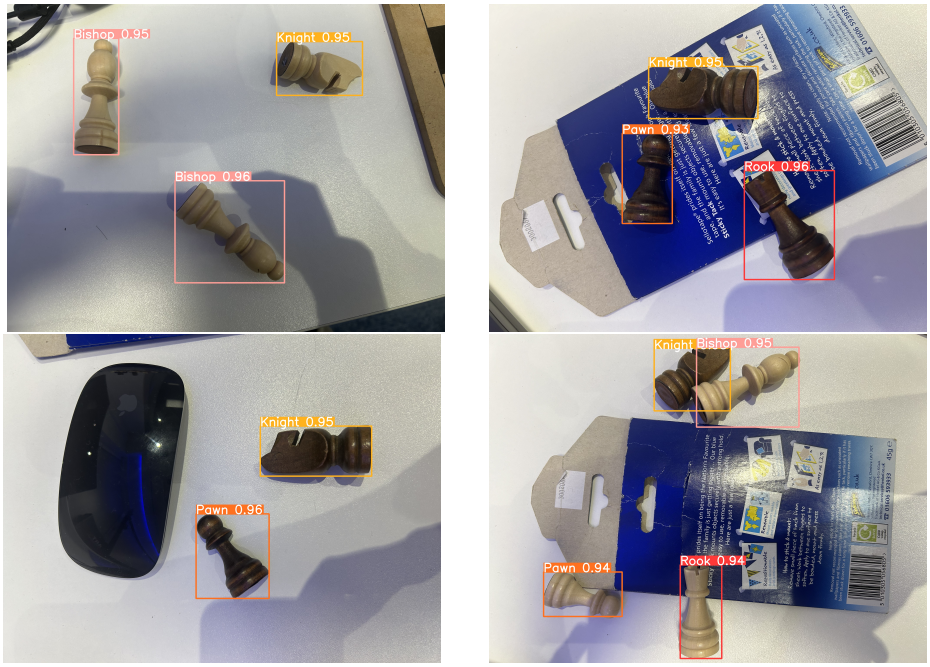

Figure 16. *Cont.*

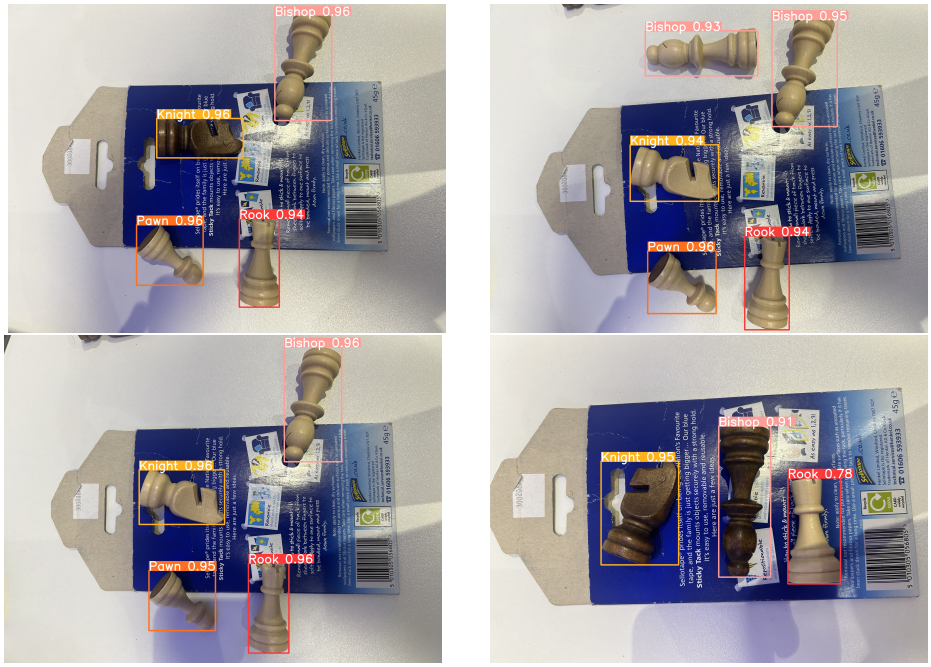

Figure 16. Actual test result graph (YOLO). The masked Knight confidence level for the first column of 4 rows is 0.94.

4.3.2. Swin Transformer Validation Dataset

In Figures 17–20, the F1 score, precision, and recall versus score threshold (=0.5), and the precision–recall curve for the classification model are presented. As can be seen from these four figures, the different classifications (rook, bishop, pawn, and knight) perform well on all metrics.

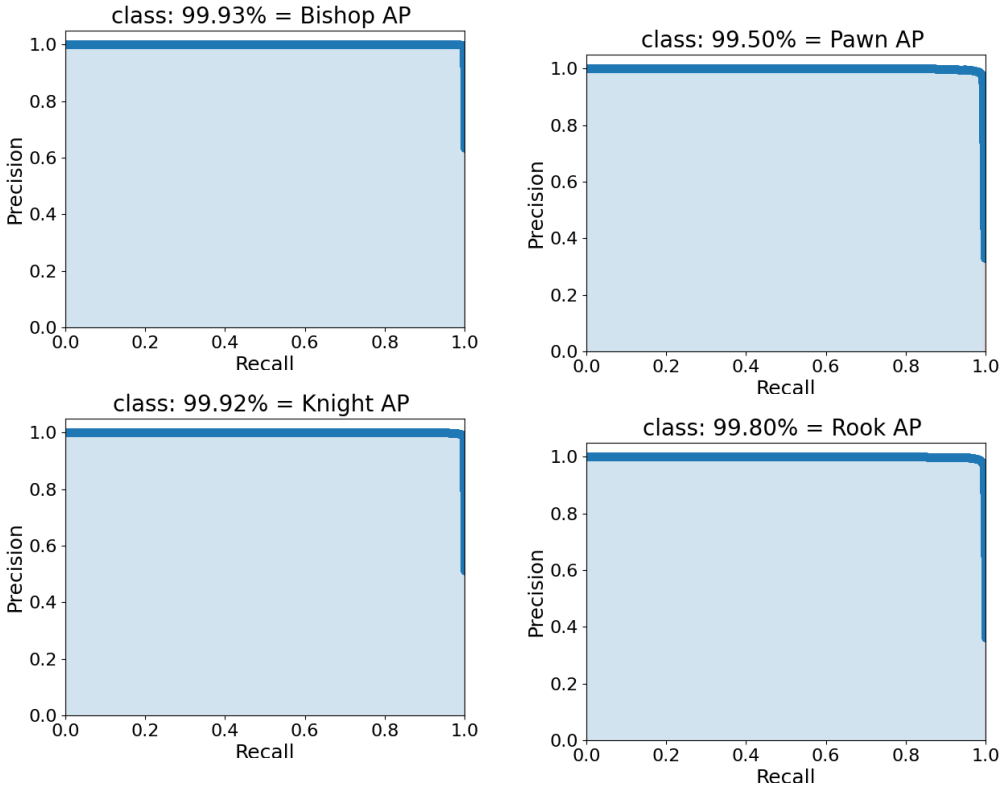

Figure 17. Curves of average precision for different categories.

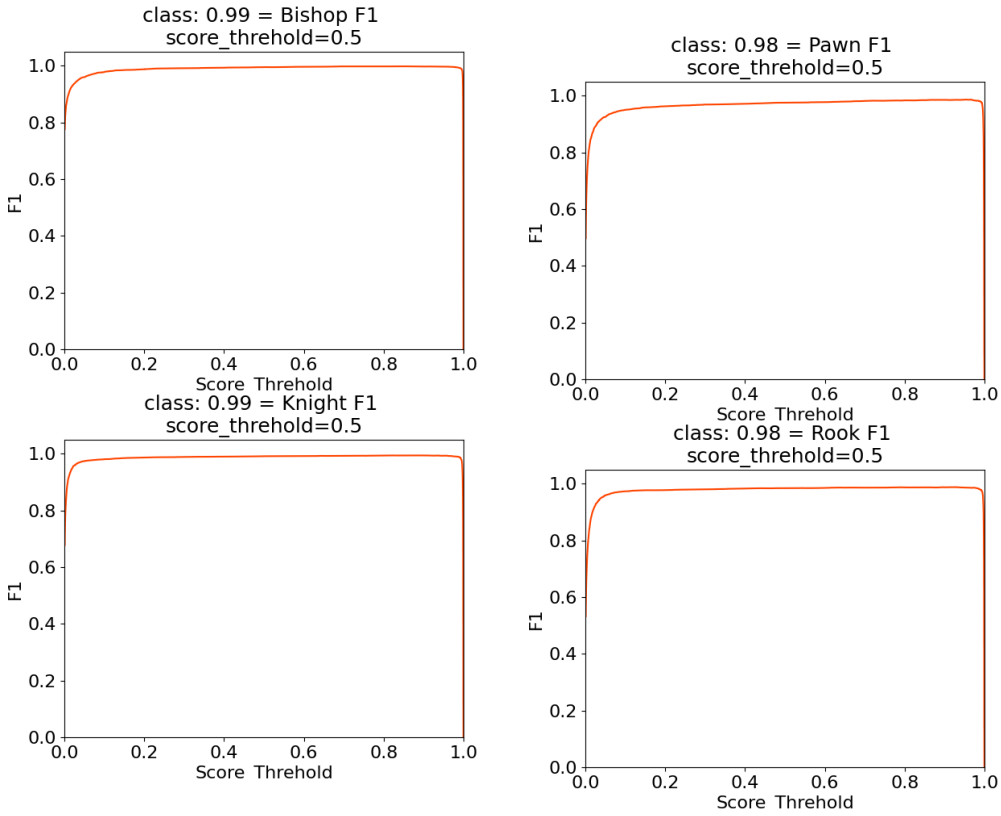

Figure 18. Curves of F1 score for different categories.

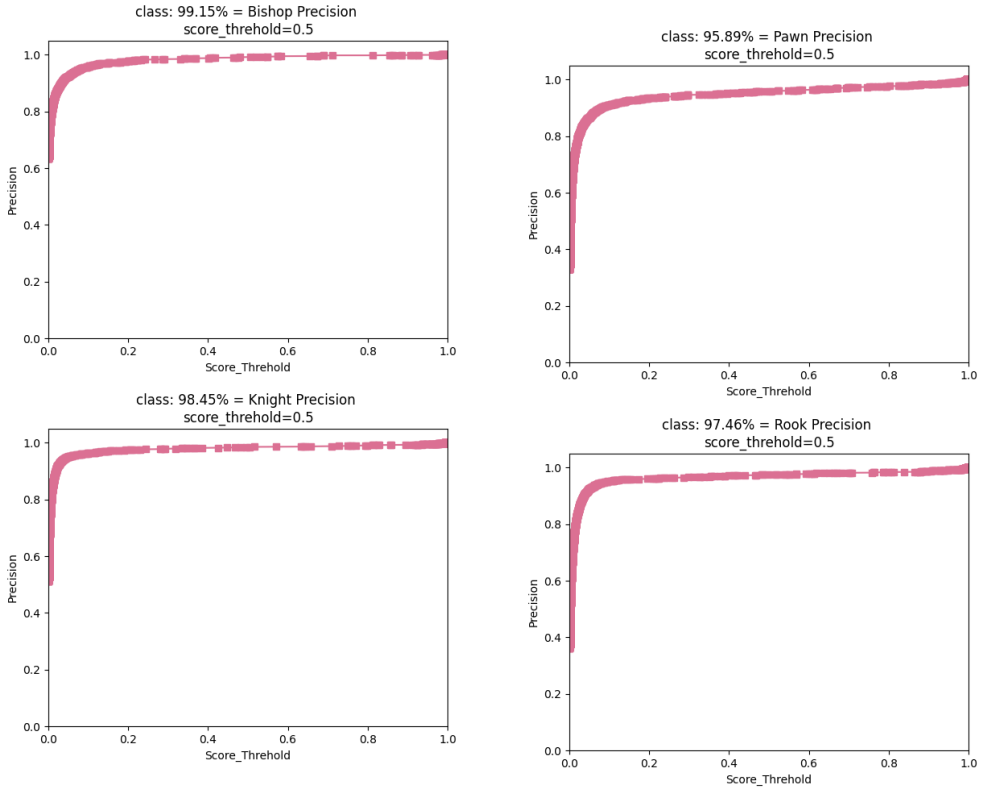

Figure 19. Curves of precision for different categories.

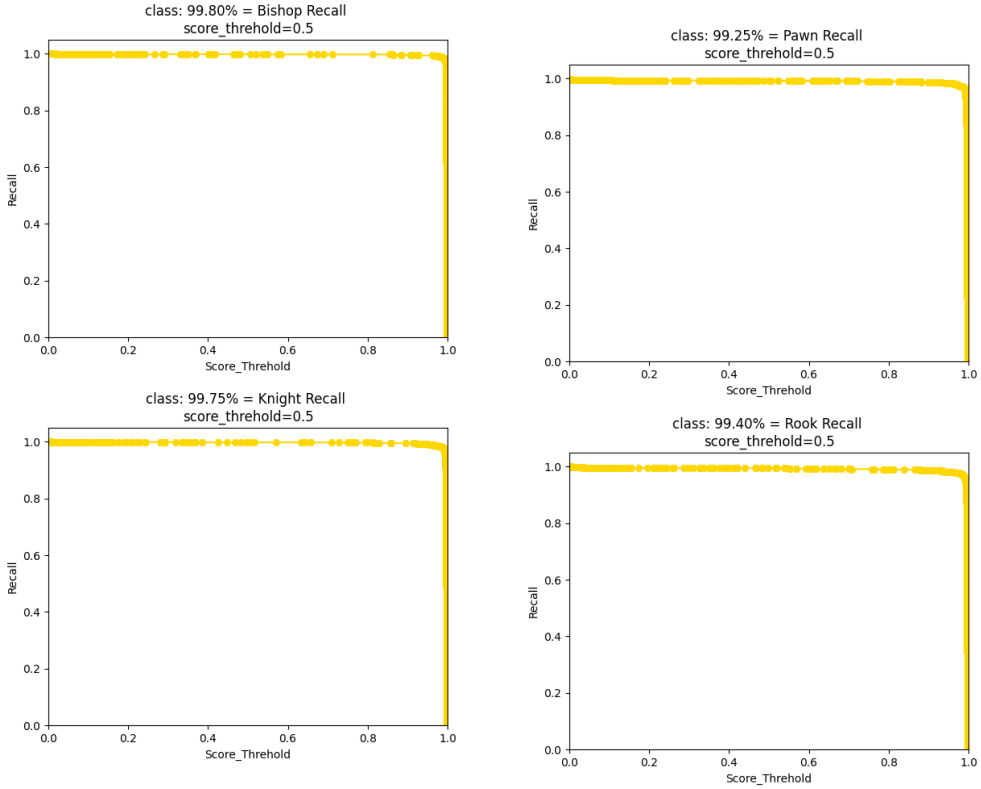

Figure 20. Curves of recall for different categories.

As shown in Figure 21, for the rook, knight, and bishop categories, the log-average miss rate is almost zero, while the pawn category has a relatively high log-average miss rate of 0.01. Figure 22 shows the distribution of object quantities across four different categories, helping to assess the representativeness and balance of each category in the dataset.

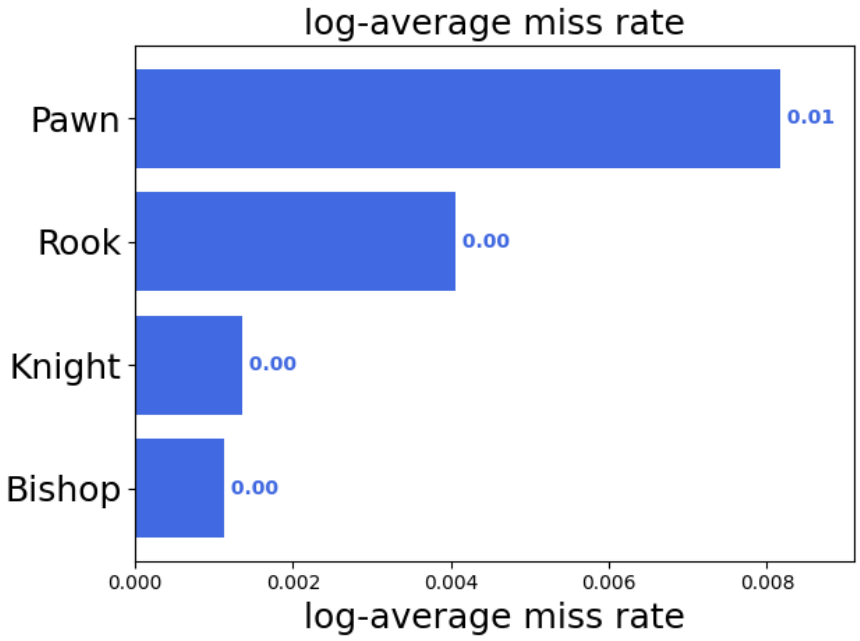

Figure 21. Log-average miss rate.

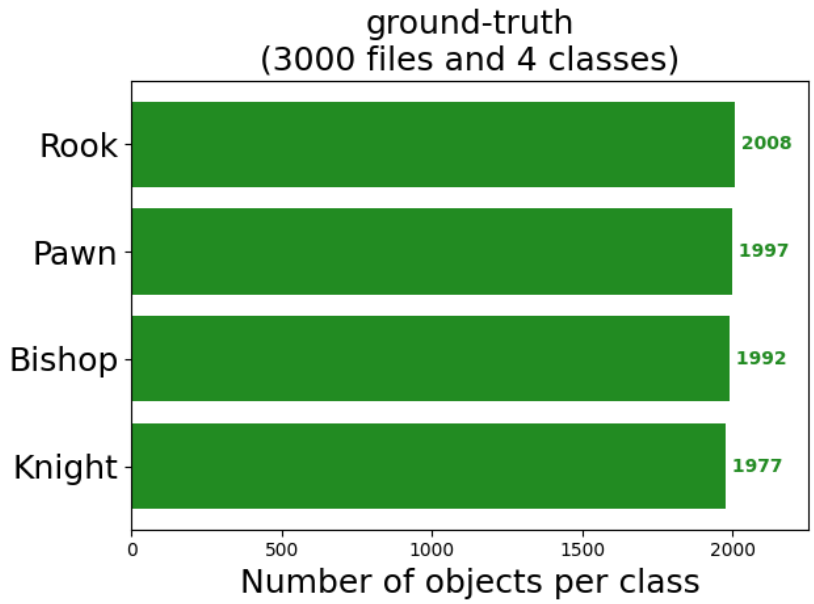

Figure 22. Ground truth information.

Figure 23 demonstrates the results of using the Swin Transformer algorithm to recognise pictures of real pieces.

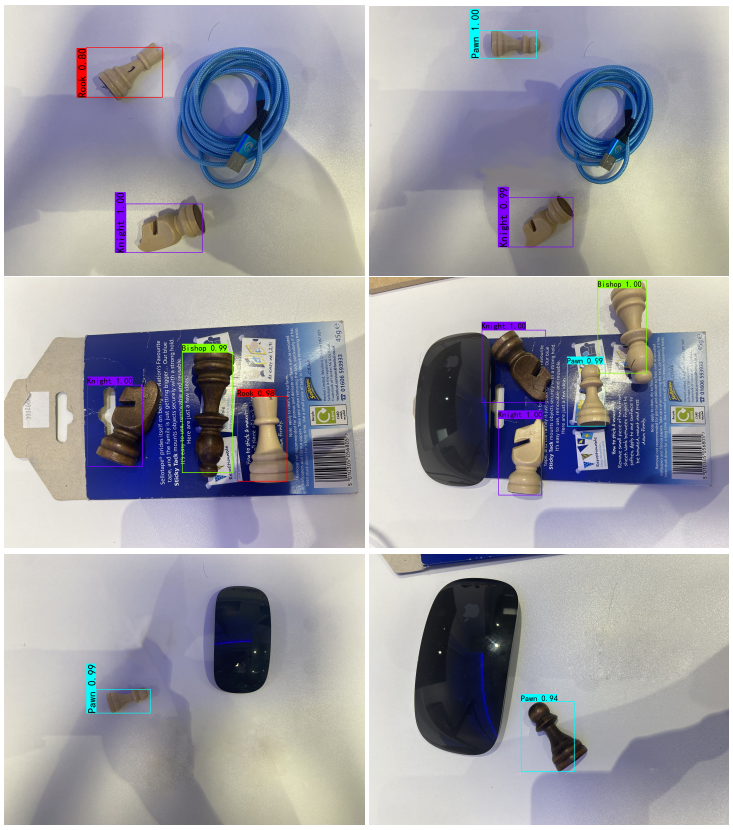

Figure 23. Actual test result graph (Swin Transformer).

5. Discussion

The main research objective of this study is to establish a virtual environment for data synthesis of selected target object models for target recognition model training. To

achieve this goal, the study was divided into three phases: Target modelling, 3D virtual environment construction, and experimental validation.

During the target modelling stage, a comprehensive literature review revealed that most virtual environment applications are quite limited and only applicable to specific scenarios. For example, SYNTHIA and the game GTA-V are primarily used for urban environments; PersonX is focused on human model research; and MINOS is applied to indoor environments. However, in practical research, the scope is often not limited to these areas, and there are often situations where digital models of target objects cannot be directly obtained. Creating realistic 3D models requires knowledge and experience with specific software, which many researchers lack. Therefore, 3D scanning was chosen for model creation. Such 3D scanning can directly extract detailed geometric information from physical objects, lowering the technical barrier and enabling faster, more efficient acquisition of high-quality 3D models.

Using 3D scanning to create target object models can result in digital models carrying reflections from the scanning process, particularly specular reflections. These unrealistic reflections can negatively impact the subsequent use of these objects in virtual environments, as they can interfere with how the objects are represented in different scenes. Therefore, once the model creation is completed, the first priority is to remove specular reflections from the model's texture, which requires balancing the efficiency and effectiveness of the algorithm. From Table 7, it can be seen that the specular highlight removal method used in this study is effective to some extent. Higher VIF values appear in image pairs 1.4 (0.8430) and 2.3 (0.8441), indicating that these image pairs retain more visual information after highlight removal. However, there are still some shortcomings in certain texture areas, such as when the highlight areas are close to other areas with similar brightness, making it difficult to remove highlights effectively, and when the colour of the highlighted portion has been bleached, making it hard to fully restore the original colour. Overall, the SSIM values are high, indicating good structural similarity after highlight removal, particularly in image pairs 2.2 (0.9983) and 2.1 (0.9982). The low MSE values in image pairs 2.1 (14.4308) and 2.3 (16.4571) indicate smaller errors between these image pairs, showing good highlight removal effectiveness. Image pairs 2.1 (36.5379) and 2.3 (35.9673) have a high PSNR, indicating lower noise and higher image quality after highlight removal. In general, the high SSIM values indicate good structural similarity, while the MSE and PSNR values show varying levels of error and noise between the image pairs. These metrics and Figures 5 and 6 indicate that the highlight removal technique used in this study can, to some extent, restore the texture details of the scanned objects.

During the literature review, it was observed that existing synthetic data engines are limited to their predefined frameworks and lack a high degree of customisation. Therefore, in this study, the establishment of the virtual environment focused on setting up basic functionalities while providing more room for customisation. Basic functions such as object positioning, background positioning, and lighting variability were implemented to allow the virtual environment to generate more diverse datasets. The decision to use Unity and the Unity Perception package was based on their powerful rendering capabilities and flexible development environment, which allow for the easy creation of realistic 3D virtual environments and the generation of high-quality synthetic data. Additionally, the Unity Perception package is specifically designed for large-scale dataset generation and annotation, enabling us to conduct experiments more efficiently and meet the research requirements.

Figure 8 demonstrates the synthetic images generated in the virtual environment. These images retained the details and texture information of the object's surface, exhibiting high visual quality. Through visual inspection, we found that the synthetic images performed excellently in terms of visual details and realism, effectively simulating real-world scenes. Figure 9 shows the results of semantic segmentation applied to the synthetic images. Supported by the synthetic data, the semantic segmentation tasks could accurately identify and classify different objects and regions within the images. Despite the synthetic images' excellent performance in detail and realism, there remains a noticeable gap in style

compared to actual photographs. These differences may be reflected in aspects such as colour rendering, lighting, and texture details, indicating that images generated in the virtual environment cannot fully replace real-world photographs in certain scenarios. These disparities warrant further research to further narrow the gap between virtual synthesis and reality.

In the data validation phase, the first step is to observe some performance metrics of the target recognition model after the training is completed. Figures 12 and 15 illustrate the variation in losses during the YOLO training process and metrics (precision, recall, and mAP). It can be observed that with the increase in training iterations, both training and validation losses gradually decrease, while model precision and recall significantly improve and stabilize. This usually indicates that the dataset does not contain excessive noise or erroneous annotations and that the data distribution is reasonable. The significant improvement and stabilisation of precision and recall suggest that the model can accurately recognise and classify different instances, reflecting that the samples in the dataset are sufficiently representative and diverse to support the model's learning. Notably, mAP@0.5 and mAP@0.5 to 0.95 on the validation set nearly reached 1.0, indicating excellent model performance across different IoU thresholds. These results demonstrate the effectiveness of using the Unity-synthesised dataset for training deep learning models. Figures 17–20 also show high values of AP, F1 score, precision, and recall on the Swin Transformer model, proving that the dataset covers sufficient variability and representative samples to allow the model to learn and generalise effectively. In addition, the dataset is not significantly noisy or mislabelled, which contributes to the accuracy and reliability of the model. Figures 14 and 22, respectively, show the number of instances of different categories (rook, bishop, pawn, knight) in the datasets used to train the two target recognition models. This indicates the frequency or number of samples for each category in the dataset. The number of objects in all categories is very close, and this balance helps the model learn the features of each category during the training process, without favouring a particular category. However, during YOLO training, it was found that the recognition model performed relatively poorly in identifying the bishop compared to other targets, so the number of bishop instances was increased separately to reflect the high degree of customisation in the virtual environment. Figure 21 shows the log-average miss rates for different categories (rook, knight, bishop, pawn). It can be seen that the log-average miss rates for the rook, knight, and bishop categories are almost zero, indicating that the model performs well in recognising these categories. The synthetic dataset provided sufficient sample diversity and feature information during training, enabling the model to accurately identify these targets. This also reflects that the synthetic dataset is of sufficiently high quality in providing balanced category samples.

The next step is the validation of the synthetic data in terms of physical dimensions and objective laws. Table 8 shows a comparison between the target object coordinates in the virtual environment and the coordinates inferred by the algorithm. The comparison reveals that the inferred coordinates are very close to the real coordinates in the virtual environment. For example, the virtual environment coordinates for the first target bishop are (0, 0, -1), while the algorithm-inferred coordinates are (0, -0.14, -0.8), showing a minimal difference. Additionally, for all target objects (bishop and knight), the inferred coordinates are highly consistent with the virtual environment coordinates. Although there are minor errors, these errors are acceptable in practical applications and may be due to the precision limitations of the model or the inherent limitations of the algorithm itself. However, the overall error range is within controllable limits, further demonstrating the effectiveness of the synthetic dataset in simulating real-world physical dimensions and objective laws. Table 9 presents the orientation vector data of the target objects from the camera's perspective. Each target object's orientation vector indicates its specific direction in the virtual environment. The orientation vector data for different objects in various positions and orientations are highly consistent, further proving the high quality and accuracy of the synthetic data.

By analysing Tables 8 and 9, the following conclusions can be drawn: The synthetic data demonstrate high precision and consistency in both position and orientation detection, confirming that these data conform to real-world physical dimensions and objective laws. The model can be effectively trained on synthetic data and shows excellent performance in testing, further validating the feasibility and effectiveness of using synthetic data for deep learning model training. These results indicate that synthetic datasets can be used not only for model training but also for simulating and testing various scenarios in practical applications, showing broad application prospects.

Figures 16 and 23 show the test results of the object recognition model trained using purely synthetic data on real images. It can be seen from the figures that the model successfully identifies and classifies multiple chess pieces (bishop, rook, pawn, knight), with generally high detection confidence. Most of the target objects have a detection confidence above 0.90, indicating that the model has high accuracy in practical applications. Moreover, even in complex backgrounds or when objects overlap with each other the model is able to simultaneously detect and classify multiple target objects. For example, in several images, the rook, pawn, and knight are detected simultaneously, and all classifications are correct. However, there are also instances where lower detection confidence can be observed. For example, in Figure 16 the model identifies the "rook" with a confidence of 0.78. This may be related to the limitations of the synthetic dataset. Although the synthetic dataset provides diverse scenes and lighting conditions during model training, it may still fall short of covering all the complex scenarios encountered in real-world applications. For instance, background text and patterns might not have been adequately represented in the training data, which could affect the model's ability to accurately delineate object boundaries.

Based on these test results, the following conclusions can be drawn: First, the performance of the object recognition model trained with purely synthetic data on real images validates its effectiveness. The high confidence detection results indicate that the model has strong recognition capabilities in real-world scenarios. However, it is also important to note that lower confidence in certain specific scenarios may reflect the limitations of the synthetic dataset. For example, the synthetic dataset might have limitations in addressing background complexity, lighting reflections, and the diversity of object positions and angles, which could affect the model's performance in these unaddressed scenarios. Nevertheless, the synthetic dataset performs well in simulating real-world object characteristics, enabling the model to learn effective features and accurately detect and classify target objects in most practical applications.

6. Conclusions

This study introduces the virtual environment based on the Unity game engine. Generating diverse, high-quality image data within the virtual environment, significantly enhances the data quality and model performance in computer vision applications. Utilising advanced randomisation techniques, this toolkit can produce rich training datasets under various lighting conditions, object positions, rotation angles, and background elements. Additionally, it supports precise distance annotations, greatly enhancing the effectiveness of the training datasets for recognition models.

The experimental results demonstrate that synthetic data generated in the virtual environment achieve notable improvements in object recognition accuracy and image clarity. Specifically, the diversity of the dataset is significantly increased through various randomisation techniques, thereby improving the model's generalisation capability and robustness across different scenarios.

This study not only showcases the enormous potential of virtual environments in generating high-quality training data but also provides important references for the processes and methods of data generation. By creating precise annotations and diverse scenes, researchers can quickly generate training datasets that meet specific requirements, significantly reducing the cost and time of dependence on real data. This efficient synthetic data

generation method is not only applicable to current research but also lays the foundation for future applications in more complex virtual scenarios.

Future Work

Despite the significant achievements of this study, there is still room for further improvement. Future work will focus on the scalability of the toolkit and explore its applications in more complex and dynamic virtual scenarios. For instance, in fields such as traffic monitoring, autonomous driving, and smart homes, by simulating complex urban traffic environments and home life scenes, the practicality and effectiveness of virtual spaces can be further validated.

Furthermore, the application of virtual spaces in other computer vision tasks, such as video analysis, multi-object tracking, and 3D reconstruction, will also be explored. By introducing time-series data and dynamic scene changes, virtual spaces will be able to generate more complex training data, aiding in the improvement of algorithms in these tasks. Additionally, combining it with other data augmentation techniques, such as generative adversarial networks (GANs), which are deep learning models where a generator and a discriminator are trained in opposition to each other to create new data that closely resemble real data, and transfer learning, can further enhance the quality and diversity of synthetic data.

In summary, this study, by developing and validating a Unity-based synthetic data virtual environment, provides an efficient, flexible, and cost-effective data generation solution for the fields of computer vision and machine learning. It offers broad prospects and possibilities for future research and applications. Moving forward, we will continue to optimise the virtual environment and explore its applications in more fields and tasks, promoting further development of virtual environments in computer vision research.

Author Contributions: Conceptualisation, C.W., L.T. and B.H.S.A.; methodology, C.W., L.T. and B.H.S.A.; resources, C.W.; writing—original draft preparation, C.W.; writing—review and editing, L.T., B.H.S.A. and C.W.; supervision, L.T. and B.H.S.A.; visualisation, L.T. and B.H.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
2. Russell, B.C.; Torralba, A.; Murphy, K.P.; Freeman, W.T. LabelMe: A Database and Web-Based Tool for Image Annotation. *Int. J. Comput. Vis.* **2008**, *77*, 157–173. [CrossRef]
3. von Ahn, L.; Dabbish, L. Labeling images with a computer game. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems CHI '04, New York, NY, USA, 24–29 April 2004; pp. 319–326. [CrossRef]
4. Ros, G.; Sellart, L.; Materzynska, J.; Vazquez, D.; Lopez, A.M. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
5. Gaidon, A.; Wang, Q.; Cabon, Y.; Vig, E. Virtual worlds as proxy for multi-object tracking analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
6. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. In Proceedings of the Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017.
7. Sun, X.; Zheng, L. Dissecting Person Re-identification from the Viewpoint of Viewpoint. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.
8. Savva, M.; Chang, A.X.; Dosovitskiy, A.; Funkhouser, T.; Koltun, V. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv* **2017**, arXiv:1712.03931.
9. Bak, S.; Carr, P.; Lalonde, J.F. Domain adaptation through synthesis for unsupervised person re-identification. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.

10. Hu, Y.T.; Chen, H.S.; Hui, K.; Huang, J.B.; Schwing, A.G. SAIL-VOS: Semantic amodal instance level video object segmentation—A synthetic dataset and baselines. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.
11. Wang, Q.; Gao, J.; Lin, W.; Yuan, Y. Learning from synthetic data for crowd counting in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.
12. Doan, A.D.; Jawaid, A.M.; Do, T.T.; Chin, T.J. G2D: From GTA to Data. *arXiv* **2018**, arXiv:1806.07381.
13. Richter, S.R.; Vineet, V.; Roth, S.; Koltun, V. Playing for data: Ground truth from computer games. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; Springer: Cham, Switzerland, 2016; Play for Data.
14. Johnson-Roberson, M.; Barto, C.; Mehta, R.; Sridhar, S.N.; Rosaen, K.; Vasudevan, R. Driving in the Matrix: Can virtual worlds replace human-generated annotations for real world tasks? In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017.
15. Fabbri, M.; Lanzi, F.; Calderara, S.; Palazzi, A.; Vezzani, R.; Cucchiara, R. Learning to detect and track visible and occluded body joints in a virtual world. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
16. Huang, Z.; Jia, Z.; Yang, J.; Kasabov, N.K. An Effective Algorithm for Specular Reflection Image Enhancement. *IEEE Access* **2021**, *9*, 154513–154523. [CrossRef]
17. Wu, Z.; Zhuang, C.; Shi, J.; Xiao, J.; Guo, J. Deep Specular Highlight Removal for Single Real-world Image. In Proceedings of the SIGGRAPH Asia 2020 Posters, ACM, Virtual, 4–13 December 2020; pp. 1–2. [CrossRef]
18. Ren, W.; Tian, J.; Tang, Y. Specular Reflection Separation With Color-Lines Constraint. *IEEE Trans. Image Process.* **2017**, *26*, 2327–2337. [CrossRef] [PubMed]
19. Wang, F.; Ainouz, S.; Petitjean, C.; Bensrhair, A. Specularity removal: A global energy minimization approach based on polarization imaging. *Comput. Vis. Image Underst.* **2017**, *158*, 31–39. [CrossRef]
20. Jo, S.; Jang, O.; Bhattacharyya, C.; Kim, M.; Lee, T.; Jang, Y.; Song, H.; Kwon, H.; Do, S.; Kim, S. S-LIGHT: Synthetic Dataset for the Separation of Diffuse and Specular Reflection Images. *Sensors* **2024**, *24*, 2286. [CrossRef] [PubMed]
21. Wu, Z.; Zhuang, C.; Shi, J.; Guo, J.; Xiao, J.; Zhang, X.; Yan, D.M. Single-Image Specular Highlight Removal via Real-World Dataset Construction. *IEEE Trans. Multimed.* **2022**, *24*, 3782–3793. [CrossRef]
22. Chang, Y.; Jung, C. Single Image Reflection Removal Using Convolutional Neural Networks. *IEEE Trans. Image Process.* **2019**, *28*, 1954–1966. [CrossRef] [PubMed]
23. Jamaludin, S.; Zainal, N. The Removal of Specular Reflection in Noisy Iris Image. *J. Telecommun. Electron. Comput. Eng.* **2016**, *8*, 59–64.
24. Jamaludin, S.; Zainal, N.; Zaki, W.M.D.W. Sub-iris Technique for Non-ideal Iris Recognition. *Arab. J. Sci. Eng.* **2018**, *43*, 7219–7228. [CrossRef]
25. Jamaludin, S.; Zainal, N.; W. Zaki, W.M.D. A fast specular reflection removal based on pixels properties method. *Bull. Electr. Eng. Inform.* **2020**, *9*, 2358–2363. [CrossRef]
26. Nie, C.; Xu, C.; Li, Z.; Chu, L.; Hu, Y. Specular Reflections Detection and Removal for Endoscopic Images Based on Brightness Classification. *Sensors* **2023**, *23*, 974. [CrossRef] [PubMed]
27. Shih, C.S.; Liao, Y.C.; Tan, C.T. Deep Learning Based End-to-End Specular Reflection Removal for Medical Endoscopic Images. In Proceedings of the International Conference on Research in Adaptive and Convergent Systems, ACM, Gdansk, Poland, 6–10 August 2023; pp. 1–9. [CrossRef]
28. Zhang, C.; Liu, Y.; Wang, K.; Tian, J. Specular highlight removal for endoscopic images using partial attention network. *Phys. Med. Biol.* **2023**, *68*, 225009. [CrossRef] [PubMed]
29. Unity Technologies. Unity User Manual. 2022. Available online: https://docs.unity3d.com/Manual/index.html (accessed on 30 May 2024).
30. Epic Games. Unreal Engine Documentation. 2022. Available online: https://docs.unrealengine.com/ (accessed on 30 May 2024).
31. Shafer, S. Using color to separate reflection components. *Color Res. Appl.* **1985**, *10*, 210–218. [CrossRef]
32. Souza, A.C.; Macedo, M.C.; Nascimento, V.P.; Oliveira, B.S. Real-Time High-Quality Specular Highlight Removal Using Efficient Pixel Clustering. In Proceedings of the 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), Paraná, Brazil, 29 October–1 November 2018; pp. 56–63. [CrossRef]
33. Artec3D. Artec Space Spider: Portable High-Precision 3D Scanning Solution. 2023. Available online: https://www.artec3d.cn (accessed on 24 June 2024).
34. Unity Technologies. Unity Perception Package. 2020. Available online: https://github.com/Unity-Technologies/com.unity.perception (accessed on 28 July 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.