

## Object classification with aggregating multiple spatial views using a machine-learning approach

Šimon Grác<sup>1</sup>, Peter Beňo<sup>1</sup>, František Duchoň<sup>2\*</sup>, Michal Malý<sup>1</sup>, Martin Dekan<sup>2</sup>

The article proposes a solution for object classification using multiple views generated from 3D data rendering and convolutional neural networks. For presentation purposes and easier verification of the solution, an application was developed to create views of 3D objects, classify them using the selected CNN, and evaluate the performance of the CNN. The evaluation is based on metrics and characteristics described in the article. Seven testing objects were used to verify the proposed solution; five CNNs were tested for each.

Keywords: 3D, OpenGL, CNN recycling, classification, classification statistics aggregation, 3D model labeling, reprojection

### 1 Introduction

In recent years, convolutional neural networks (CNNs) have made significant strides in enhancing the speed and accuracy of object classification. Despite these notable achievements, there are situations in which these networks falter, especially when dealing with objects presented from various angles or positions. In such instances, integrating a 3D scene model proves invaluable, as it facilitates the generation of diverse 2D perspectives of the same scene. Subsequently, employing CNNs for object classification based on these perspectives emerges as an effective strategy.

When addressing this issue, it is unnecessary to construct a custom CNN architecture from scratch. Instead, it is recommended to seek pre-trained networks specifically tailored for this task. These networks, known as classification networks, include well-established architectures such as Google's Inception, renowned for their remarkable balance between performance and computational demands [1]. Another successful recent entrant is ResNet, which demonstrates comparable results [2] [3]. YOLO is another architecture we considered, owing to its prowess in object classification, particularly its detection speed [4, 5, 16]. Given that our approach independently generates 3D data, we can tailor the considered objects to align with the selected network's training. The effectiveness of our proposed solution is evaluated based on the predicted object occurrence percentage in the image.

Notably, CNNs are not exclusively employed for object detection but also for object segmentation, i.e.,

isolating objects from their backgrounds. Noteworthy architectures designed to address this challenge include FCNs [6, 7, 19] and Mask R-CNN [8-11, 20]. This work strives to assess the performance and adaptability of networks, aiming to select the most suitable one for classifying specific objects. It involves considering variations of the same networks or networks trained on different objects.

This article centres on harnessing three-dimensional data to enhance the reuse of existing and newly developed machine-learning procedures. Typically, these procedures excel in controlled conditions, where images are captured under ideal lighting without extraneous disturbances. Under such circumstances, the input image can be predicted, simplifying the neural networks' task. Our proposed solution employs a 3D model of an object that a trained neural network endeavours to locate in its 2D representation. We leverage a 3D scene model in conjunction with CNNs to bolster the reliability of object classification. Specifically, we demonstrate that aggregating data from multiple 2D perspectives within a 3D scene model yields higher object classification accuracy. This approach permits the "recycling" of existing neural networks, trained on extensive datasets, to classify objects from various viewpoints. Furthermore, labeling objects based on aggregated statistics derived from numerous perspectives instills greater confidence than relying on a single viewpoint.

Moreover, the paper presents experimental findings illustrating the substantial enhancements in object classification reliability achievable through our

<sup>1</sup> Photoneo, s.r.o, Plynárenská 6, Bratislava, Slovak Republic

<sup>2</sup> Slovak University of Technology, Ilkovičova 3, Bratislava, Slovak Republic

\* frantisek.duchon@stuba.sk

approach. Our research reveals that aggregating labels from diverse views enables identifying and rectifying errors in object classification. Based on these results, we conclude that label aggregation within a 3D scene model presents a promising avenue for augmenting the reliability of object classification using convolutional neural networks.

The presented research is industry-oriented on the typical application of bin picking [18] with industrial robots. Created point clouds in bin-picking applications analyse the optimal selection of an object that the robot can pick from the bin. It is especially possible for objects for which there is a CAD model. However, in the industry, many objects do not have CAD models. We focus our research on such objects. The proposed approach offers the possibility of effectively using existing CNNs to classify objects in a 2D scene on created point clouds, which are commonly used in bin-picking applications. Thus, this approach makes using significantly successful classifiers (2D images + CNN) more efficient for identification in point clouds without creating new structures of CNNs and new datasets [15, 17].

The article structure comprises five sections. The initial section outlines the procedure for acquiring 3D

models of selected objects. The second section elucidates the method for representing these 3D models in 2D space. The third section deliberates on the selection of CNNs subjected to testing. In the fourth section, the methodology for evaluating the experiments detailed in the fifth section is expounded upon.

## 2 Methods of obtaining 3D models of selected objects

PhoXi 3D Scanner S [12] was used to create a point cloud representing the surface of the scanned objects. The overall scene was created using the Meshlab tool. The 3D object generated exhibits areas devoid of data. This absence of data is attributed to the fact that these regions were concealed from the sensor's view due to other sections of the object obstructing them. Consequently, capturing the object from diverse angles is recommended to address this limitation. A rotary table (Fig. 1) was used to achieve such capturing. When set in motion, this rotary platform systematically produces a predetermined quantity of 3D images, contingent upon the desired fidelity of the final model. The halting of the rotary platform is carefully synchronized with the object scanning process.



**Fig. 1.** Rotary table and 3D model of the scene

The resulting 3D model begins with a collection of generated point clouds and information regarding the transformations between camera views. The output is a 3D model of the object in mesh format. This multifaceted process comprises the following key steps:

1. Alignment: The alignment of input scanned data into a unified coordinate system.

2. ICP Algorithm: Utilizing a chosen ICP algorithm to minimize the distances between individual scan points. It involves finding the nearest neighbors for each point in one scan within the next scan and minimizing the distances between these points.

3. Data Filtering: The filtration of data that does not pertain to the scene, often caused by noise, reflections, and other undesired artifacts. It involves verifying the presence of processed points from one scan in subsequent scans.

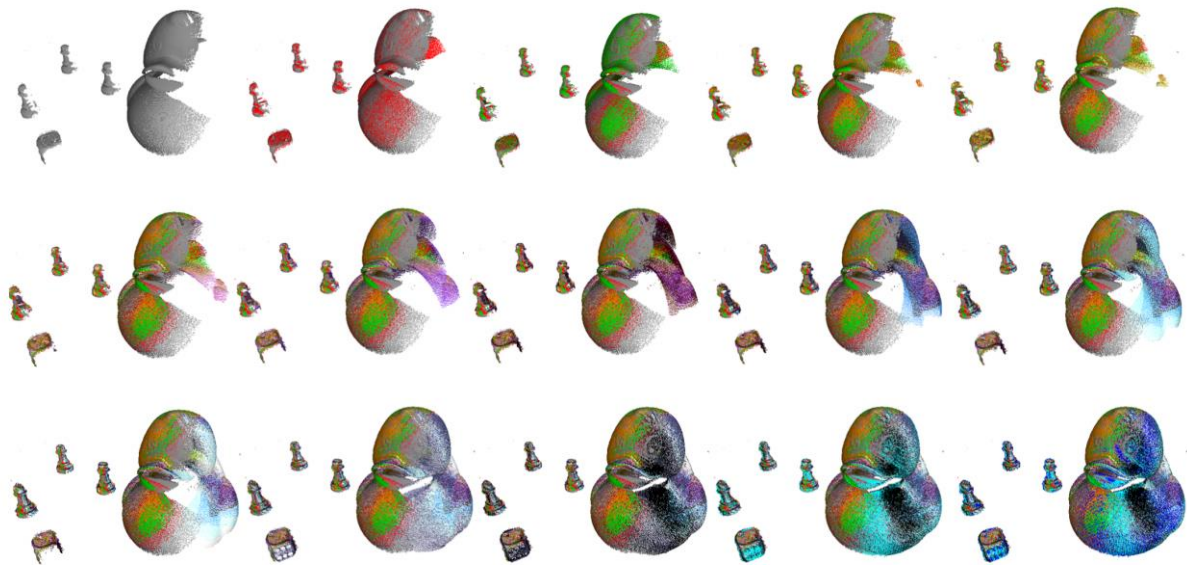
4. Unification and Surface Reconstruction: Integrating the scans into a cohesive whole and performing

surface reconstruction. It can be achieved through techniques like Poisson reconstruction or other methods that approximate a function based on the input points defining the object's surface.

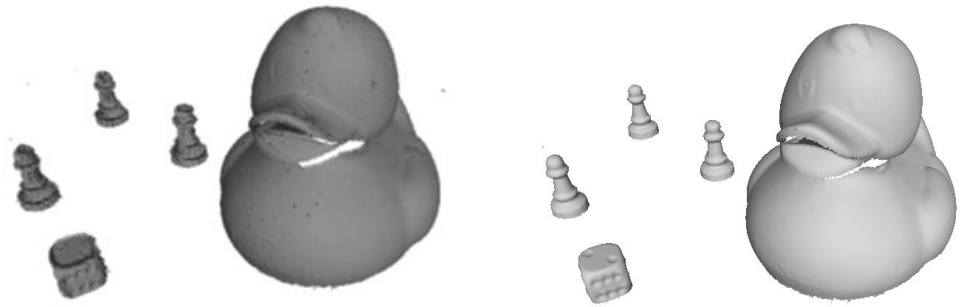
5. Triangular Mesh: Obtaining the final triangular mesh using a chosen algorithm for triangulation, such as Marching cubes or Tetrahedra.

The process of amalgamating multiple scans is visually represented in Fig. 2, where each new scan is denoted in a distinct color. The ultimate 3D model is constructed from a total of 15 scans. The resulting point cloud is depicted in Fig. 3. Still, it necessitates further refinement, encompassing the removal of erroneous data, surface reconstruction, and the eventual acquisition of the triangular mesh, as shown in the figure.

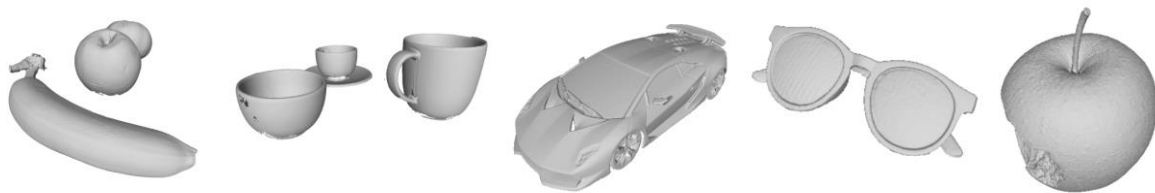
Subsequently, a series of 3D models are generated, which will serve as the basis for generating views and subsequent classification using CNN. Examples of these generated 3D models are showcased in Fig. 4.



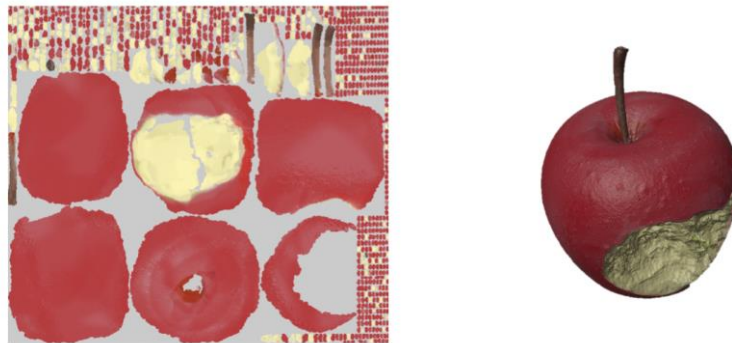
**Fig. 2.** The process of merging scans. The data from each new scan are depicted in a different colour on the image, and the final object consists of a total of 15 scans (the first scan is in the top left corner, the last scan is in the bottom right). The resulting scene displays 3 game figurines, a dice, and a rubber duck.



**Fig. 3.** Resulting point cloud (left) and mesh (right)



**Fig. 4.** Example of created 3D models



**Fig. 5.** The texture used to color the 3D model and the resulting colored 3D model

### 3 Generating 2D views of 3D object models

We employed an OpenGL environment to generate 2D views of 3D object models, configuring the following settings:

- Lighting: A direct white light source was directed above the object.
- Background: A white background was utilized.
- Projection Matrix: The projection matrix featured perspective image settings with fixed image cropping parameters. It defined the distances on the z-axis from which the projection is considered.

- Model Matrix: A fixed model matrix was employed.
- Image Matrix: A gradual transformation of the image matrix was executed to simulate the camera's rotation around the object.

To generate views of the 3D model, we moved the virtual camera along an imaginary sphere. The point 'P,' situated on the sphere's surface, is characterized by three parameters:

- Radial Distance (r): Representing the distance of point 'P' from the fixed starting point 'S,' located at the sphere's center.



- Polar Angle ( $\theta$ ): This angle originates from the center of the sphere and defines the deviation angle of the position vector  $\vec{r}$  considering the x-axis.
- Azimuth Angle ( $\varphi$ ): This angle reflects the angle between the positional vector  $\vec{r}$  of point 'P' and the z-axis, with the vertex at the sphere's center.

The camera is consistently adjusted backward by a predetermined value to ensure the scene is visible. This displacement value was determined experimentally and corresponds to the radial distance 'r' from the object's center in the scene. The range of rotation for individual angles is specified as follows:

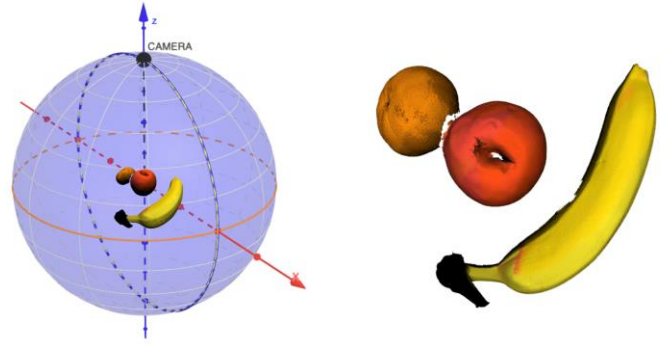
- Rotation around the z-axis produces 60 views, corresponding to a  $6^\circ$  increment in the angle  $\varphi$ .
- Along the x-axis, the camera's movement is confined between  $0^\circ$  and  $180^\circ$  to prevent the generation of identical views with a rotated object. The camera traverses this axis seven times, considering positions at  $0^\circ$  and  $180^\circ$ , altering the angle  $\theta$  by  $30^\circ$ .

The initial camera position, denoted as  $C_i$ , is defined by coordinates  $(\varphi, \theta) = (0^\circ, 0^\circ)$ . In this position, each 3D model is viewed from an overhead perspective (as illustrated in Fig. 6).

In generating views, the OpenGL library was harnessed, allowing various projection matrices to be applied. The projection matrix is pivotal in determining which parts of the scene are displayed in 2D and which are omitted. It defines the mapping of 3D coordinates to 2D screen coordinates. We tested two types of projection matrices: perspective and orthographic. For both, it's imperative to specify six parameters establishing two cropping planes. In orthographic projection, the center of projection is at infinity, and the projection direction remains consistent and perpendicular to the image surface for all scene points. The orthographic projection matrix,  $M_o$ , is defined as:

$$M_o = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1)$$

In the provided context, the notation is as follows: 'l' represents the cropping from the left side, 'r' signifies the cropping from the right side, 't' corresponds to cropping from the top, 'b' denotes cropping from the bottom, 'n' designates the depth on the z-axis from which the projection is taken into account, and 'f' specifies the depth on the z-axis up to which the projection is considered.



**Fig. 6.** Initial camera position and the generated view from this camera position

The projection process, common to both orthographic and perspective types, unfolds in two fundamental stages:

1. Conversion from Eye Coordinates to Clip Coordinates: This initial step involves transforming all object points from eye coordinates into clip coordinates, resulting from multiplying the model matrix with object coordinates. These clip coordinates are achieved by multiplying the projection matrix with the eye coordinates.

2. Transformation into Normalized Device Coordinates (NDC): Subsequently, the clip coordinates undergo conversion into normalized device coordinates (NDC). This transformation is accomplished by dividing by the component 'w' [13].

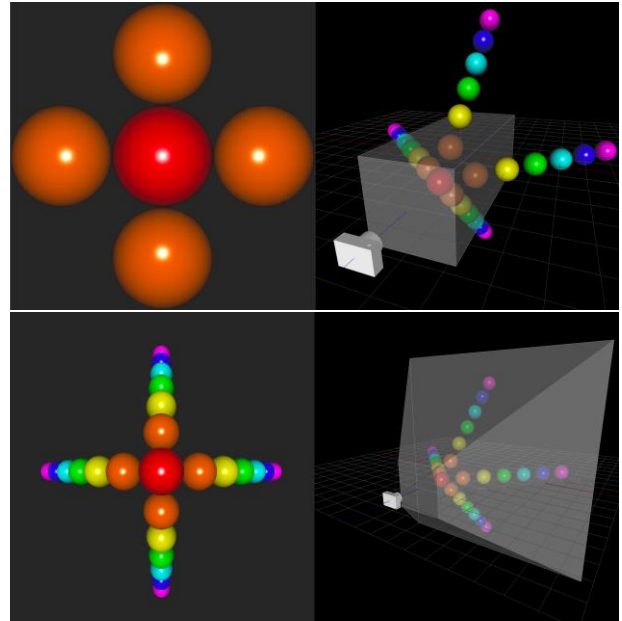
The perspective matrix, distinctive for its ability to depict scenes with a more lifelike perspective, accounts for the varying sizes of objects or the scene itself, contingent on the camera's distance from them. This perspective projection is described by matrix  $M_p$  designed as follows:

$$M_p = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-(f+n)}{f-n} & \frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad (2)$$

The symbols in Eqn. (2) have the same meaning as those in Eqn. (1). A visual juxtaposition of orthographic and perspective projections is illustrated in Fig. 7.

#### 4 Sets of CNNs

We leveraged the Keras library to discern the most fitting neural network (NN) structure, benefiting from 18 CNNs available for unrestricted application. These networks are equipped to classify a wide array of object categories, having undergone extensive training on the ImageNet image database, which comprises over 1.4 million images. In addition to these 18 networks, our testing suite encompassed three additional CNN networks, resulting in 21 CNNs under scrutiny (as presented in Table 1). Two networks used the COCO (Common Objects in COntext) database, boasting a vast image collection of 2.5 million.



**Fig. 7.** Comparison of orthographic projection (left) and perspective projection (right) in the OpenGL

**Table 1.** Networks that can be tested in the created application

Name of the network	Supported number of classes	Image dataset	Number of parameters	Depth
YOLO	80	COCO	-	-
Mask_RCNN	80	COCO	-	-
Xception	1 000	ImageNet	22 910 480	160
VGG16	1 000	ImageNet	138 357 544	-
VGG19	1 000	ImageNet	143 667 240	-
Resnet50	1 000	ImageNet	25 636 712	-
Resnet101	1 000	ImageNet	44 707 176	-
Resnet152	1 000	ImageNet	60 419 944	-
Resnet50V2	1 000	ImageNet	25 613 800	-
Resnet101V2	1 000	ImageNet	44 675 560	-
Resnet152V2	1 000	ImageNet	60 380 648	-
Inception	21 000	ImageNet	-	-
InceptionV3	1 000	ImageNet	23 851 784	159
InceptionResNetV2	1 000	ImageNet	55 873 736	572
MobileNet	1 000	ImageNet	4 253 864	88
MobileNetV2	1 000	ImageNet	3 538 984	88
DenseNet121	1 000	ImageNet	8 062 504	121
DenseNet169	1 000	ImageNet	14 307 880	169
DenseNet201	1 000	ImageNet	20 242 984	201
NasNetMobile	1 000	ImageNet	5 326 716	-
NasNetLarge	1 000	ImageNet	88 949 818	-

## 5 Evaluation metric

Various methodologies were employed to assess the performance of the chosen CNNs, a common practice for evaluating data classification. This evaluation occurred on the test set after each network completed the training and validation phases.

To initiate the fundamental assessment, a classic contingency table comprising four values [14] was utilized: TP (true positive), TN (true negative), FP (false positive), and FN (false negative). While the proposed evaluation system categorizes objects into multiple classes, the success measurement is confined to a binary classification perspective, focusing solely on the considered object within the 3D model. The overall success rate is quantified as the ratio of correct classifications to the total number of classifications [14]:

$$acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

The true positive rate (TPR) and the rate of incorrectly classified positive samples (FPR – false positive rates) will be used to express the success of CNN classifiers. Sensitivity (TPR) is determined as follows:

$$TPR = \frac{TP}{TP + FN} \quad (4)$$

The inverse value of sensitivity expresses the rate of incorrectly classified negative samples (False Negative Rate):

$$FNR = \frac{FN}{TP + FN} \quad (5)$$

Specificity defines the rate of correctly classified negative samples (True Negative Rate):

$$TNR = \frac{TN}{FP + TN} \quad (6)$$

The inverse value representing incorrectly classified negative samples (False Positive Rate) is derived from it:

$$FPR = \frac{FP}{FP + TN} \quad (7)$$

Each value will be calculated concerning the acceptance threshold, forming classification curves. More details about these classification curves can be found in [14].

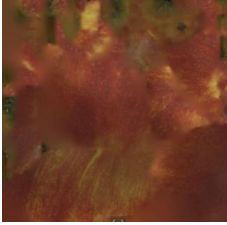
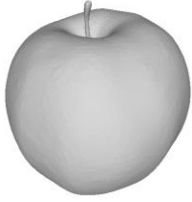

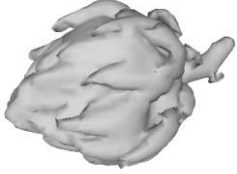
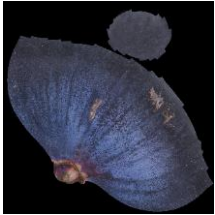

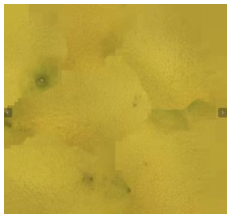
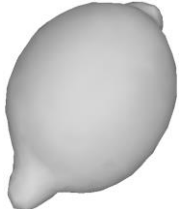


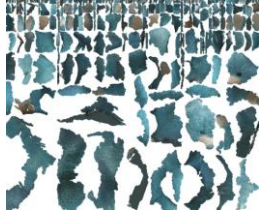

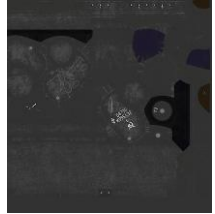
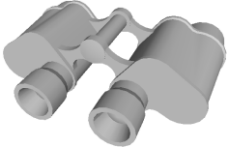
## 6 Results

To assess the efficacy of our proposed solution, we employed a combination of 3D models generated using the PhoXi 3D scanner and readily available models sourced from the Internet (as detailed in Table 2). Throughout the application's development, we conducted experiments involving varying scene and object settings, with optimal outcomes achieved when adhering to the following guidelines:

- Ensuring the considered object is scaled appropriately to maximize its presence within the generated view.
- Implement rotations that avoid interference between the object and unobservable camera areas.
- Attaining scene lighting synchronized with the camera's position, ensuring direct illumination of the object without casting shadows.

To streamline the experiments, we focused on scenes featuring a solitary object. This approach simplified the evaluation process, as handling multiple objects in the scene would complicate the identification of "bad views" based on the established conditions. Moreover, evaluating the network's performance in scenarios with multiple objects would offer limited informative value. In cases where an object was obscured by another object or incorrectly captured by the camera, such instances indicated the network's inability to classify the object. Consequently, we manually segmented the objects from the created scenes, submitting only these isolated objects to the network for classification.

**Table 2.** Set of tested objects and selected parameters

Object	Max. spherical angle (°)	Classes of negative images	Texture preview	Preview of the 3D object
1 – apple	180 (420 views)	Pear, peach, apricot, pomegranate, tomato		
2 – artichoke	180 (420 views)	Sugar apple, donut, green strawberry, banana, gyromitra		
3 – fig	180 (420 views)	Plum, purple grapes, blackberries, red onion, blueberry		
4 – lemon	180 (420 views)	Tangerine, grapefruit, orange, lime, Granny Smith apple		
5 – banana	90 (240 views)	Rocket, worm, corn, sea cucumber		
6 – Teddy bear	120 (300 views)	Brown bear, polar bear, raccoon, gorilla		
7 - binoculars	180 (420 views)	Buckle, lighter, microphone, holster, bottle		



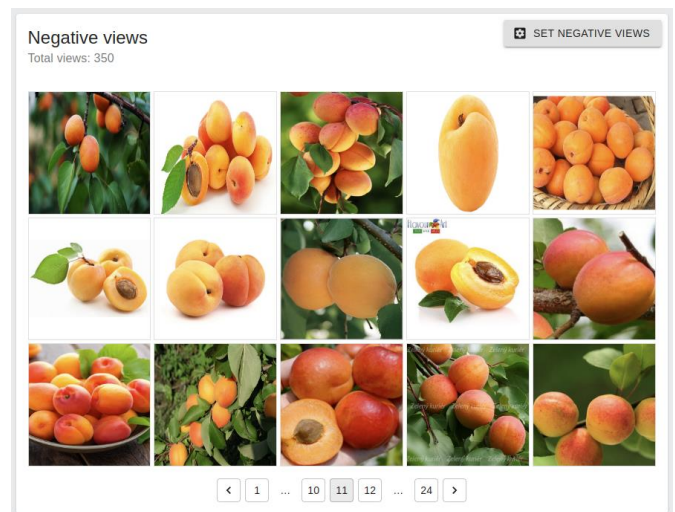
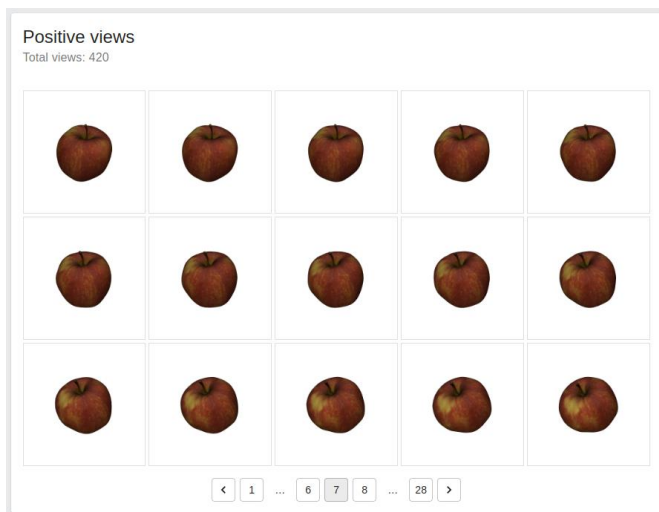
The experiments were executed as follows:

- A series of views was generated for the imported 3D object, spanning from 0° to the maximum spherical angle, and a texture was applied to these views.
- Negative images depicting objects listed in Table 2 were generated. The selection of these objects was based on a combination of objects supported by the networks and objects that exhibited similarity when subjected to object classification testing.
- The classification process was initiated for each model, comprising both the generated views and the negative images, using multiple networks that support the classification of the specific object.
- The resulting label for each model from these classifications was determined based on the label that occurred most frequently.
- Subsequently, the selected CNNs were compared, and the most successful one was determined based on the experimental outcomes.

### 6.1 Object class apple

Apple object detection is supported by Mask\_RCNN and YOLO networks. Other networks support the classification of specific apple varieties, so only these two were used in testing. The scanned object is complete, so views were generated up to the maximum possible spherical angle of 180°. The generated views of the object are shown in Fig. 8. The views of the negative images used to test the network's success are shown in Fig. 9.

The tested networks achieved a high success rate (Table 3). The resulting labeling of the 3D model (the most frequently occurring label) is correct for both networks. The YOLO network achieved an overall success rate (the success rate of correctly classifying the generated views and correctly rejecting negative images) higher than the Mask\_RCNN network. Incorrectly classified generated views for the YOLO network were only 4. The network achieved the best results with an acceptance threshold value of 0.95.



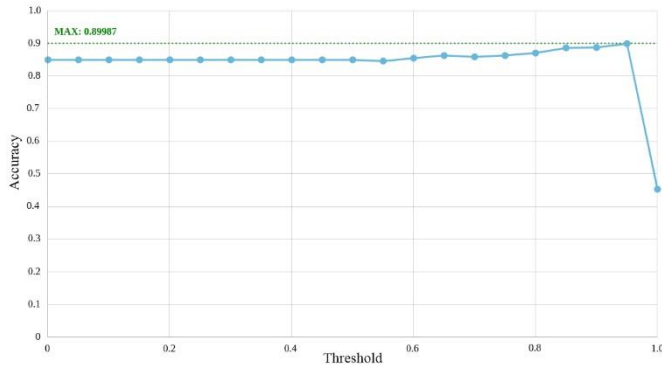
**Fig. 8.** Example of several out of 420 generated 2D views of the scanned 3D point cloud of the Apple class.

**Fig. 9.** The set of negative images for the class Apple included images downloaded from the internet such as peaches, apricots, oranges, etc., with a total count of 350.

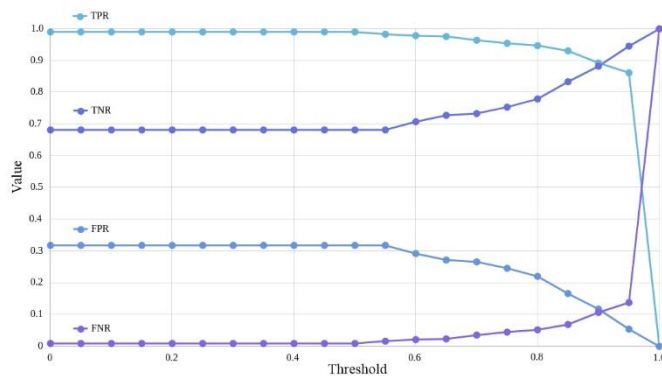
**Table 3.** Object class apple – classification results

Name of the network	Overall success rate (%)	Classification success of generated views (%)	Acceptance threshold at maximum success	The most common labeling
YOLO	90.0	99.0	0.95	apple
Mask_RCNN	74.5	76.9	0.95	apple

The accuracy achieved, depending on the change in the acceptance threshold of the YOLO network, is shown in Fig. 10. The maximum success achieved in this case was 89.99%. The development of TPR, FPR, TNR, and FNR metrics, depending on the threshold change, is shown in Fig. 11.



**Fig. 10.** The achieved accuracy of the apple object classification by the YOLO network depending on the acceptance threshold (The achieved maximum accuracy for the threshold range from 0 to 1.0 is depicted in green color, while the results for individual thresholds with a change of +0.04 are indicated in blue.)



**Fig. 11.** The values of individual metrics TPR, FPR, TNR, and FNR for the classification of objects of type "Apple" by the YOLO network, their dependence is shown based on the choice of threshold type. The change in threshold values is scaled by +0.04.

From the results, it can be concluded that the YOLO network is suitable and recyclable for classifying an object (3D model) of the Apple class. The network labeled almost all views correctly, and the overall success rates are good. Also, TPR and TNR reach high

values, and FNR and FPR, on the contrary, have low values.

### 6.2 Object class artichoke

All networks from the Keras library support artichoke object detection. Five networks were randomly selected (Table 4). The best success rate was achieved by the Inception V3 network, with a total success rate of 90.3% and a success rate of classification of generated views of 81.4%. Seventy-eight generated views out of a total of 420 were incorrectly classified. Examples of unsuccessfully classified views are shown in Fig. 15. Artichoke was often classified as a banana or a mushroom in unsuccessful attempts. The network performed best with a very low acceptance threshold.

### 6.3 Object class fig

The fig class object is also supported by all networks from the Keras library, and five networks from this library was also randomly selected to verify detection from generated views and selected negative images (Table 5). The most successful network, in this case, was ResNet152V2, which correctly classified 50.7% of views and achieved a total success rate of 67.65%. Up to 207 out of 420 generated views were incorrectly classified. However, even with many incorrect classifications, the labeling was correct for all tested networks.

### 6.4 Object class lemon

All networks from the Keras library also support the lemon class object, and five networks from this library were randomly selected (Table 6). DenseNet169 achieved the highest overall success rate, and NASNetLarge achieved the best classification rate of generated views. All networks were able to classify the object successfully.

### 6.5 Object class banana

The banana class object is supported by all networks, from which five were randomly selected for testing (Table 7). In this case, the model is scanned from above and incomplete, so views were generated only up to a polar angle of 90°, or a total of 240 views. DenseNet201 networks achieved the highest success at a low acceptance threshold of 0.2 and Yolo at a threshold of 0.55. Both networks achieved an overall success rate of approximately 85% and a classification success of

generated views of 70%. Thus, approximately 70 views were incorrectly classified. Most of these views come from views generated at a polar angle of 90°, where numerous imperfections of the generated 3D model are visible.

#### 6.6 Object class Teddy

All Keras library networks support the teddy bear class object. Five networks were randomly selected and tested with 300 generated views of this object (Table 8). The highest classification success was achieved by the Xception network, which successfully classified 42% of views, representing up to 173 incorrectly labeled views (out of 300). This result corresponds to the observed variety of views when the object resembles a teddy bear,

mostly only from the front side of the object. Other networks achieved very low classification success rates, but most of them could still determine the resulting labeling of the 3D model successfully.

#### 6.7 Object class binoculars

All networks from the Keras library support the binoculars class object, and five were randomly selected for testing (Table 9). All the selected networks could determine the final labeling of the 3D model correctly. The InceptionV3 network achieved the highest success but with a low classification acceptance threshold. For example, with a threshold of 0.7, the overall success rate is 72.8%. The number of incorrectly classified views is 75 out of 420.

**Table 4.** Artichoke object classification results

Name of the network	Overall success rate (%)	Classification success of generated views (%)	Acceptance threshold at maximum success	The most common labeling
MobileNet	85.1	71.0	0.2	artichoke
ResNet50	79.0	58.8	0.15	artichoke
Vgg16	86.5	73.8	0.2	artichoke
InceptionV3	90.3	81.4	0.05	artichoke
Xception	78.7	58.6	0.1	artichoke

**Table 5.** Fig object classification results

Name of the network	Overall success rate (%)	Classification success of generated views (%)	Acceptance threshold at maximum success	The most common labeling
InceptionV3	59.4	35.5	0.45	fig
Xception	65.31	48.8	0.4	fig
DenseNet169	64.32	49.8	0.55	fig
ResNe152V2	67.65	50.7	0.7	fig
MobileNetV2	58.27	31.7	0.1	fig

**Table 6.** Lemon object classification results

Name of the network	Overall success rate (%)	Classification success of generated views (%)	Acceptance threshold at maximum success	The most common labeling
DenseNet169	93.5	98.1	0.75	lemon
NASNetLarge	92.6	98.8	0.9	lemon
MobileNet	65.8	49.3	0.05	lemon
ResNet152	88.9	91.9	0.3	lemon
Xception	91.3	98.3	0.2	lemon

**Table 7.** Banana object classification results

Name of the network	Overall success rate (%)	Classification success of generated views (%)	Acceptance threshold at maximum success	The most common labeling
YOLO	84.1	70.0	0.55	banana
ResNet152V2	78.4	57.9	0.35	banana
InceptionV3	82.0	65.0	0.1	banana
DesneNet201	86.3	72.9	0.2	banana
MobileNetV2	79.3	59.2	0.1	banana

**Table 8.** Teddy object classification results

Name of the network	Overall success rate [%]	Classification success of generated views [%]	Acceptance threshold at maximum success	The most common labeling
MobileNet	51.8	11.3	0.1	hook
ResNet101V2	57.7	22.7	0.2	teddy
Vgg19	52.5	12.0	0.05	necklace
Xception	68.2	42.3	0.05	teddy
NasNetMobile	61.3	29.0	0.05	teddy

**Table 9.** Binoculars object classification results

Name of the network	Overall success rate (%)	Classification success of generated views (%)	Acceptance threshold at maximum success	The most common labeling
ResNet50V2	73.8	51.0	0.25	binoculars
DenseNet201	73.6	50.2	0.1	binoculars
Vgg16	65.1	34.3	0.1	binoculars
InceptionV3	90.5	82.1	0.05	binoculars
MobileNetV2	80.7	63.6	0.05	binoculars

## 6.8 General comments on the results

Drawing from the obtained results, several overarching observations can be made:

- Most classifications exhibit notably low False Positive Rate (FPR) values, reflecting their proficiency in minimizing misclassifications.
- Many classifications demonstrate high True Negative Rate (TNR) values, highlighting their effectiveness in correctly identifying non-object instances.
- The most informative metrics for recognizing a specific object are True Positive Rate (TPR) and False Negative Rate (FNR).
- TPR and FNR are pivotal in determining how effectively a multi-classifier can handle binary classification tasks.
- The classification acceptance threshold tends to be lower for networks designed to support the classification of numerous objects.

When assessing whether a selected network can accurately classify the chosen object, a critical prerequisite is that the network has been trained on a class representing such an object. Our experiments aimed to ascertain the extent to which the selected classifier could discern the presence or absence of the object in the generated view. In this context, the most crucial metrics are TPR and FNR, which gauge the accuracy of classifying the view as "object" or "non-object" within the given acceptance threshold. The acceptance threshold's value is intrinsically linked to the classifier's capability to make correct classifications while rejecting negative images. It also influences the accurate selection of the classes represented in the negative images.

Our experiments revealed that networks achieve their highest success rates at lower classification acceptance thresholds. Notably, this tendency can be partly attributed to numerous objects within the classifier. A

direct comparison of threshold values for maximum classification success between X-class networks (comprising 80 classes) and Y-class networks (encompassing 1,000 classes) underscores this trend:

- X-type networks achieve their best results at higher thresholds (0.95, 0.95, 0.55).
- In contrast, Y-type networks attain optimal results at lower thresholds (ranging from 0.05 to 0.35).

When the acceptance threshold is increased for Y-type networks, we observe a decrease in the rate of accurate classifications, leading to a lower overall success rate. This behavior is particularly noticeable when negative images are predominantly recognized correctly and are appropriately assigned to their corresponding classes by the network. In such cases, there is a scenario in which negative images are consistently identified correctly, and even a small portion of the object within the generated view is sufficient to warrant a successful classification. Consequently, most views are accurately identified, and most negative images are rejected even when using the minimum classification acceptance threshold. Conversely, when negative images are poorly identified, meaning they are erroneously labeled with the class of the 3D model, a higher classification acceptance threshold becomes necessary.

It implies that if we know the object classes that the selected network can effectively recognize, we can introduce them as negative images. If the network rejects these negative images, we can confidently employ a lower acceptance threshold. In this scenario, where erroneous classifications are infrequent, opting for a lower threshold will yield a higher classification success rate than selecting a higher threshold, leading to a considerable loss of correct classifications.

In certain cases, the False Positive Rate (FPR) metric reaches a zero value because the negative images do not fall within the class of the considered object. It signifies that in the case of object A classification and a negative image B, the classifier either labels image B as B (if it can classify such an object) or selects another class from its supported classes that is most akin to object B. The likelihood of the classifier correctly identifying the class to which the object under consideration belongs from a large pool of supported classes is quite low. This phenomenon is especially prevalent in networks that support the recognition of numerous classes. The reason behind the frequently occurring high value of the True Negative Rate (TNR) metric is analogous, as it is the reciprocal value of the FPR metric.

We postulate that a similar scenario might arise if we attempt to recognize multiple objects within a scene. In this context, we would only cover a portion of the

supported classes when utilizing negative images. However, this hypothesis warrants further investigation.

The aggregation of classifications from multiple spatial perspectives significantly aids in the final object labeling, even when dealing with networks that may not be ideally suited for detecting the specific object due to their achieved success rate. While the number of correctly classified views may be less than 50%, this still proves adequate for achieving correct overall labeling. In many instances, even networks that were not highly successful managed to classify the correct class as the second or third choice. Consequently, during the final object labeling, it is prudent to consider classifications with the highest probability and others. This approach enables a statistically more sophisticated assessment of the classification data.

## 7 Conclusion

Out of 32 classifications spanning seven distinct objects, our chosen networks accurately classified the object within the 3D model 30 times. This high success rate, achieved through aggregating the most frequently occurring labels, equates to a remarkable 96% precision in object labeling. It's worth noting that model labeling encountered challenges in the case of the teddy bear object.

Our testing affirms that our solution reliably delivers the final labeling of 3D models by automating the generation of views and subsequent classification using existing convolutional neural networks. Importantly, our proposed methodology doesn't gauge the overall success of the selected network but rather elucidates its competence in determining the presence or absence of the specific object within the generated views and negative images. Furthermore, we've demonstrated that the desired outcome can be attained by aggregating information from multiple perspectives, even when working with networks not originally trained for such classifications.

The aim of this article was to highlight the possibilities of using CNNs trained on 2D image data for the classification of 3D point clouds. The method of obtaining the point cloud scans is irrelevant for the results presented in the article, as only one of many possible methods is demonstrated. Another approach could involve placing the camera on the end effector of a robot. An important aspect of the design is how to segment individual objects into different views and how to aggregate the results from these views. The results indicate that object identification would achieve high accuracy even with incomplete point clouds. However, this was not the goal of this research.



## Acknowledgement

This publication was created with grants KEGA 028STU-4/2022, DIROZ, and AIR (AI in Robotics, excellent teams). Company Photoneo also supported this research.

## References

- [1] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *2016 IEEE conference on computer vision and pattern recognition*, 2016.
- [2] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *2017 AAAI conference on artificial intelligence*, 2017.
- [3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE conference on computer vision and pattern recognition*, 2016.
- [4] K. He and J. Sun, "Convolutional neural networks at constrained time cost," *2015 IEEE conference on computer vision and pattern recognition*, 2015.
- [5] J. Redmon, S. Divvala, G. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE conference on computer vision and pattern recognition*, 2016.
- [6] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *2015 IEEE conference on computer vision and pattern recognition*, 2015.
- [7] R. Hu, M. Rohrbach, and T. Darrell, "Segmentation from natural language expressions," *2016 Computer Vision—ECCV 2016: 14th European Conference*, 2016.
- [8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE conference on computer vision and pattern recognition*, 2014.
- [9] R. Girshick, "Fast r-cnn," *2015 IEEE international conference on computer vision*, 2015.
- [10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks". *Advances in neural information processing systems*, vol. 28, 2015.
- [11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *2017 IEEE international conference on computer vision*, 2017.
- [12] PhoXi 3D Scanner S. Available online: <https://www.photoneo.com/products/phoxi-scan-s/>
- [13] S. Ahn, OpenGL Projection Matrix. Available online: [http://www.songho.ca/opengl/gl\\_projectionmatrix.html](http://www.songho.ca/opengl/gl_projectionmatrix.html)
- [14] A. Tharwat, "Classification assessment methods," *Applied computing and informatics*, vol. 17, no. 1, pp. 168-192, 2020.
- [15] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," *IEEE/CVF Conference on computer vision and pattern recognition*, 2019.
- [16] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.
- [17] H. Wang, S. Dong, S. Shi, A. Li, J. Li, Z. Li, and L. Wang, "Cagroup3d: Class-aware grouping for 3d object detection on point clouds," *Advances in neural information processing systems*, vol. 35, 2022.
- [18] L. D. Hanh and K. T. G. Hieu, "3D matching by combining CAD model and computer vision for autonomous bin picking," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, Vol. 15, pp. 239-247, 2021.
- [19] J. Wang, L. Song, Z. Li, H. Sun, J. Sun, and N. Zheng, "End-to-end object detection with fully convolutional network," *2021 IEEE/CVF conference on computer vision and pattern recognition*, 2021.
- [20] X. Xu, M. Zhao, P. Shi, R. Ren, X. He, X. Wei, and H. Yang, "Crack detection and comparison study based on faster R-CNN and mask R-CNN," *Sensors*, vol. 22, no. 3 pp. 1215, 2022.

Received 14 February 2024

---