3D-Adv: Black-Box Adversarial Attacks against Deep Learning Models through 3D Sensors

Kaichen Yang*, Xuan-Yi Lin†, Yixin Sun‡, Tsung-Yi Ho†, and Yier Jin*
*University of Florida, †National Tsing Hua University, ‡Huazhong University of Science and Technology
bojanykc@ufl.edu, s108062544@m108.nthu.edu.tw, 121929989@qq.com, tyho@cs.nthu.edu.tw, yier.jin@ece.ufl.edu

***Abstract*—The combination of deep learning techniques and commercial 3D sensors reveal a bright future as they provide a low cost and convenient method to collect and analyze depth information from the environment for various applications ranging from industrial modeling to mobile face recognition. Despite the abundant research devoted to the development of more accurate, flexible and efficient machine learning schemes as well as 3D sensors, security concerns related to these techniques remain largely untouched. In this paper, we propose a novel adversarial attack against this combination by showing that deep learning models with popular 3D sensors may misclassify real objects in the physical environment. Comparing to the existing attack algorithms against deep learning models developed for 3D data analysis that only consider digital point cloud data and single deep learning model, our attacks target popular commercial 3D sensors combined with various deep learning schemes in the black-box setting. Experimental results demonstrate that our 3D printed adversarial objects stay effective after scanned by a 3D sensor.**

I. Introduction

With the popularization of various 3D sensors including Lidar, scanner and depth camera, the cost to collect depth information from the environment reduces dramatically. The demand to process the massive 3D data collected by these sensors calls for an efficient and accurate method, where deep learning techniques come in. As many deep learning models are proposed to process 3D data such as [1]–[5], it is popular to apply deep learning models for analyzing the 3D data for classification, detection, and recognition. For instance, depth information from human faces captured by the structured-light-based depth camera set on smartphones can be utilized by the deep learning models within these smartphones for accurate face recognition.

Though the combination of deep learning and 3D sensors is viewed as a promising way in many applications, the security concerns toward them are also raised, yet the efforts invested in the related security problems are quite limited. The vulnerability of deep learning models in image processing and voice recognition are already revealed in [6], [7], as well as in domains beyond image and voice [8], [9], showing that adversarial examples generated by adding small but intentionally selected perturbations to the original inputs can (mis-)lead the target models to specific incorrect outputs. This type of attack is also verified in physical world through specific devices. As shown in [10], [11], adversaries can achieve their purpose against deep learning models behind the devices by printing images captured by camera or broadcasting songs targeting smart home devices.

978-1-6654-3274-0/21/$31.00 ©2021 IEEE

Recent works [12] start to consider adversarial examples in 3D data, i.e., point cloud. Algorithms in [12], [13] target the PointNet series model [2], [3] by adding or altering the point cloud served as the inputs to the model. However, the ignoring of realistic scenarios with specific types of 3D sensors weakens the feasibility of these attacks. We do observe that in the autonomous driving area, many works start to consider attack schemes against the combinations of deep learning models and Lidar sensors [14]. However, similar works are rarely discussed in applications related to other 3D sensors such as 3D cameras and scanners. The work in [15] considers the attack on PointNet model and 3D sensors simultaneously. However, the white-box setting gives the work less flexibility and thus not suitable for commercial devices and real world scenarios. We also notice that new algorithms of 3D data classification algorithms [4], [5] emerge after PointNet series models [2], [3], yet attack methods are less discussed on those models.

To better understanding the security issues related to the combination of 3D sensors and deep learning models, we focus on adversarial attacks in black-box scenario against this popular combination. A novel attack method is proposed to generate robust adversarial 3D objects that preserve their properties after physically manufactured by 3D printers. The attack is verified in popular 3D data classification models available for testing.

To summarize, we make the following contributions in this paper:

- We propose a novel attack approach to generate 3D adversarial objects against structured-light-based 3D sensors with popular deep learning-based 3D data classification models;
- In our experiments, we show that our attack stay effective against various types of deep learning structures designed for analyzing point cloud data;
- We demonstrate that our attack can be launched without knowing the internal information of the target models during the attack and can be generalized to multiple types of deep learning models in 3D classification domain.

The remainder of this paper is organized as follows: Section II describes related work. Our proposed method is introduced in Section III. Experiments are performed in Section IV, and Section V concludes this paper.

II. Related work

A. Commercial 3D Vision System

Popular 3D vision system can be generally divided into two categories: passive imaging and active imaging.

Passive imaging. Stereo imaging system is a popular passive 3D capturing solution. It involves two separate cameras which take images of a scene from two different viewpoints. The disparity derived from the two paired images help compute the relative depth of points in the scene, where 3D information can be generated. Though passive imaging methods can be implemented easily, they are limited by the accuracy and the capability of capturing small and mobile objects.

Active imaging. Structured light is a popular active imaging method. It employs structured laser light or two-dimensional laser grid projection. The projected coherent laser beams hit the surface of the object, generate patterns that can be captured by the camera. With the calibration information which contains the geometry of the camera and laser combination, the coordinates of the intersection point between the surface and the laser beam can be calculated by triangulation. Popular 3D scanners normally require calibration and placing the scanned objects in a static base to acquire high-accuracy point cloud data from the object surface. This type of 3D scanners are widely applied in real-world applications such as designing and manufacturing.

ToF (Time of flight) camera is another type of 3D sensors. It continuously sends light pulses to the target, and then using the sensor to receive the light returned from the object, the distance of the target is obtained by detecting the flight (round trip) time of these light pulses. ToF cameras have advantages such as fast processing speed, comparatively longer range and compact design. Since it is a relatively new technique, solutions based on ToF are not mature yet. Due to these constraints, the more mature structured light based 3D scanner will be chosen as the targeted sensors in our work.

B. Deep Learning on 3D Data

As 3D sensors get popular, the demand to process 3D data collected in an efficient and accurate way rises. Point cloud data, as a popular data format that contains 3D coordinates information of points sampled from the surface of physical or virtual objects, are widely applied in many areas including industrial modeling, surveying, and autonomous driving. Though deep learning models are used to process point cloud data, the different characteristics between point cloud data and common image data make it difficult to directly apply existing deep learning techniques. Various methods are developed to transform the point cloud data into 3D grids or features that can be analyzed by existing deep learning models [1], but the information loss during the transformation limits their performance. PointNet and PointNet++ [2], [3] are developed to address this limitation by directly processing the raw point cloud data in the deep learning models and enabling end-to-end neural network learning. PointNet is robust against missing points and random perturbations. The high performance of PointNet leads to its wide adoption in different applications as backbone feature generation networks. Since then, more deep learning schemes [4], [5] are proposed to directly analyze the point cloud data.

C. Adversarial Attacks in Deep Learning

The phenomenon of adversarial examples was first found by Szegedy [6], who observed that adding slight but intentionally generated perturbations to legal inputs can mislead deep learning models into making incorrect decisions in 2D image classification. The problem to generate adversarial examples is often generalized as an optimization process to find a perturbation on the original input x so that the target deep learning model can give the specified output. Since then, more algorithms [16]–[18] have been proposed to launch increasingly efficient and effective adversarial attacks in different domains. These algorithms are all white-box attacks. They rely on the transferability [19] to stay effective when the internal information of the target model is not accessible. Black-box adversarial attacks such as Zeroth Order Optimization (ZOO) [20] do not rely on transferability, but rather depends on the output of the target models, and performing optimization with gradient estimates obtained via finite differences. Generative adversarial networks (GAN) and genetic algorithm [21] are also applied in generating adversarial examples in black-box settings. However, how to generate reasonable adversarial meshes against deep learning models in the 3D analyzing domain remains an open question.

III. Proposed Methodology

A. Threat Model

Our work assumes an adversary who intends to attack the combination of deep learning models and structured-light-based 3D cameras by manufacturing certain objects in the physical world. We select popular models designed for point cloud classification as the target, including PointNet++ [3], PointConv [5], and PCNN [4]. The printed adversarial objects will be scanned by the 3D sensors and the point cloud generated from the corresponding depth map will be verified whether the model can be attacked physically. The attack will be launched in the black-box model due to the following concerns: the gradients required for white-box attacks are inaccessible within the sensors; in realistic scenarios, adversary may find it hard to acquire the internal information in the models.

B. Genetic Adversarial Attack in 3D Sensing

Genetic algorithm. The proposed black-box attack relies on genetic algorithms, which are population-based gradient-free optimization strategies. Our work is inspired by [21], which first integrates genetic algorithms into black-box adversarial attacks against deep learning models. Genetic algorithm is a general approach that requires to define gene, population, fitness function, mutation and crossover to simulate the evolving process in the nature selection. The population of inputs generated from the genes are evolved through mutation and

crossover to maximize the fitness score, which evaluate the performance of each candidates regarding the goal of evolving. At every iteration, the candidate with the highest fitness score is preserved while the rest are replaced. New candidates are generated by mutating and crossover a pair of old candidates. In our attack algorithms, we define a single gene as a mesh consists of vertices and faces and the population is a set of genes. The gene could be either retrieved from existing datasets (e.g., ModelNet40), or generated from the variants of unit icosphere by adding small Gaussian noise. The population will be scored by the fitness function at the beginning of each iteration and retain the superior portion of genes in the population. Afterward, the mutation function is performed on parents by pre-defined mutation chance except the best gene. The crossover step will then take place to restore the population size by randomly combined genes in parents to generate new gene, as known as child, and add it into the population.

Fitness function. The fitness objective follows the algorithm proposed by Carlini and Wagner [18], including the following objectives: A C&W-like attack objective regarding the classification results which motivate the population to be classified as certain object type by the target deep learning models.

The gene, which is the vertices of a mesh, can be represented as $x \in \mathbb{R}^{n\times 3}$. The perturbation added to vertices during the mutation is represented as $\in \mathbb{R}^{n\times 3}$. The classification objective $f(x)$ is formulated as follows:

$$f(x +) = \max_{i \neq y'} fZ(x +)_i g \quad Z(x +)_{y'} \tag{1}$$

where y' is the target class, $Z(\ )$ denotes the output tensor of the model, which is the logit score before the softmax function in the last layer. To limit the distances (dis-similarity) between the perturbed examples and the original one, we use Chamfer distance to find the point p in the mesh x and its closest point p' in corresponding x' to measure the average distance of points, which is introduced as $D_C(x'; x)$ in Equation 2:

$$D_C(x'; x) = \frac{1}{kx'k} \sum_{p' \in x'} \left(\min_{p \in x} kp' \quad pk_2^2 \right) \tag{2}$$

In contrast to other types of data, e.g., images, the values of vertex coordinate in meshes are unbounded, causing that mutations of vertices may alter the candidate mesh into an unstable and twisted form. To avoid the cases that meshes are evolved into shapeless, we introduce composite distance metric l_{com}, which help the mesh to stay in a reasonable shape. The proposed composite distance metric consists of three terms, Laplacian loss l_{lap}, edge loss l_{edge} and normal loss l_{nor} of the evolving meshes [22]. The classification objective term l_{cw} is defined in Equation 1.

The overall fitness function f is shown below.

$$f(x +) = \quad (l_{cw} + c \quad D_C(x + \ ; x) + l_{com}) \tag{3}$$

$$l_{com} = !_1 \quad l_{lap} + !_2 \quad l_{edge} + !_3 \quad l_{nor} \tag{4}$$

where $!_1$, $!_2$, and $!_3$ are coefficients. We set $!_1 = 0{:}1$, $!_2 = 1$ and $!_3 = 0{:}01$. The hyper-parameter c is the weight for the applied distance matrices to limited the deformation of the mesh. The number of iteration is set to 4000 unless early stop criteria are reached.

Mutation. For each iteration of the genetic attack, we retain the top n genes of the population with the highest fitness scores. Those genes with lower scores which are not in top n is called leftover and will be retained with probability p_1. Afterward, those genes except the best one will be mutated with probability p_2 by the following mutation function.

$$f(x) = x + M \quad (N(0; 1) \quad) \tag{5}$$

where $x \in \mathbb{R}^{n\times 3}$ is the coordinate of the mesh vertices and M is a mask matrix. The mask matrix can be consider as a selection of vertex that should be perturbed by adding a Gaussian distribution with a scalar . The denotes the element-wised multiplication, which applies the mask M on the perturbation. The is a user-defined coefficient which limits the perturbed value to prevent the mesh from excessive distortion.

Crossover. After the mutation step, we need to restore the population size by randomly crossover the existing genes in the population. In this step, two genes will be randomly picked, which are called parents, from the population and combined using a mask to obtain half of the gene from both parents to form a new gene, known as a child. This process will continue until the population size is restored. The crossover function is defined as follows.

$$f(x_1; x_2) = M \quad x_1 + \overline{M} \quad x_2 \tag{6}$$

where x_1 and x_2 are the parent genes. The mask M is used for selecting vertices from parent genes. The overall attack process is depicted in Algorithm 1.

Attack types. Two types of attacks are considered in our work, Reshaping Attack and Altering Attack. In the Reshaping Attack, we will reshape a standard sphere so that it can be classified as any target class by the target deep learning model. In the Altering Attack we will try to slightly alter the existing mesh belong to certain category, e.g., "bottle", so that it can be classified as certain class, e.g., "vase".

IV. Experiments

A. Experimental Setup

The ModelNet40 dataset [23] is used in our experiments, including training, testing the victim models, and generating adversarial examples. This dataset contains 12,311 CAD models with 40 common object categories in real world. We use the default splits, where 9,843 examples are used for training, and the remaining 2,468 examples are used for testing. For adversarial attacks, we randomly choose a subset of data in both training and testing set as the "clean" meshes.

PointNet++ [3], PointConv [5], and PCNN [4] are chosen as our target models. The models are trained using the ModelNet40 dataset with network architectures and hyper-parameters used in original papers. In our experiments, these

Algorithm 1: Genetic attack algorithm

Input: mesh m, fitness function $Fit()$, mutation function $Mutate()$, crossover function $Cross()$, max-iteration T, probability for selecting leftover p_1, probability for mutation p_2, population size P;
Output: Adversarial mesh m_{adv};
initialization;
Gene vertices and faces of mesh;
Population $\{\text{Gene}_i\}_{i=1}^{P}$;
for i 1 **to** T **do**
  Fit(Population);
  parents best n genes;
  leftovers genes parents;
  foreach *gene leftovers* **do**
    parents.insert(gene) with probability p_1;
  end
  foreach *gene 2 parents and gene best gene* **do**
    $Mutate$(gene) with probability p_2;
  end
  while *size of parents < population size* **do**
    child $Cross$(random genes in parents);
    parents.insert(child)
  end
  if *early stop or successfully generated* m_{adv} **then**
    return m_{adv} best gene
  end
end

classification models are trained without considering normal vectors to classify 40 categories.

In our work, Reshaping Attack starts from a sphere to an arbitrary class and the most-likely Altering Attack aiming to misclassifying the ModelNet40 mesh. We pick a subset from the ModelNet40 including bottle, bed, door, cone, lamp, and vase. For each class, we choose 15 objects and perform the proposed genetic attack. Specifically, we sample the point from the mesh surface by PyTorch3D [22] to obtain the point cloud for the fitness function in each iteration. We then use 3D printers and 3D scanners to show that our method can be implemented in the real world.

All experiments were carried out on a server with an Intel E5-2623 v4 2.60GHz CPU with 16GB RAM, Ubuntu 18.04, accelerated by NVIDIA CUDA Framework 10.0 and cuDNN 7.0 with four NVIDIA GeForce RTX 2080Ti GPUs. The adversarial objects are printed by Creality CR-10 Max 3D Printer. The printed 3D objects are then scanned as meshes by EinScan-SE 3D scanner. The re-scanned meshes will be served as the inputs to the target models to test the effectiveness of the physical adversarial examples.

B. Most-likely Altering Attack on ModelNet40 Mesh

We select a subset of the ModelNet40 and apply our proposed attack method. The class with the second-highest prediction value is selected as our target class to perform the most-likely attack. Since the shape and the structure can vary from different classes, in order to make the attack successful, larger distortion and more iterations are needed. Thus, we choose the most-likely class to generate a more regular adversarial example. The visualized results and the attack success rate are shown in Figure 1 and Table I, respectively. Depending on the class, the success rate is between 60% 100%. From Table I, we can observe that "bottle" and "door" achieve higher success rate than the others. This is due to the required distortion for those classes to transform into another relatively close classes is less than others. For instance, door and curtain are similar in shape and thus more likely to attack successfully. This result indicated that our proposed method can be even more effective on those models trained for classifying specific types of data. From Figure 1 we can observe that the generated adversarial objects stay reasonable shape so that they can be physically manufactured/printed. Note that existing works such as [12], [13] generate adversarial point cloud data by adding unattached point cluster without forming watertight mesh. The attack presented in [15] forms adversarial meshes as well, yet their methods rely on the surface reconstruction algorithm provided by third-party software, which may cause unintentional detail loss.

TABLE I
ATTACK SUCCESS RATE (%) OF EACH CLASS IN ALTERING ATTACK.

Model \ class	bottle	cone	lamp	bed	door	vase
PointNet++	93.33	60	60	73.33	100	73.33
PointConv	93.33	66.67	73.33	60	86.67	60
PCNN	93.33	66.67	80	66.67	86.67	60

C. Reshaping Attack from Standard Sphere

The Reshaping Attack is launched through reshaping the standard sphere. Hence, it can be classified as any target class in the ModelNet40 dataset while maintaining a reasonable shape to be realized physically. The success rates of the Reshaping Attack regarding different types of target models are shown in Table II. It shows the number of classes successfully attacked, the total number of classes in ModelNet40, and the mis-classification rate. The reshaping attack achieve success rate between 87:5% 92:5%. The higher success rate comparing to Altering Attack can be attribute to the less limitation of Chamfer distance.

The visualization of the original standard sphere, some adversarial objects as well as the target model and classes are shown in Figure 2. We can observe that through our attack a standard sphere mesh can be reshaped into watertight mesh according to the purpose of the adversary against various types of deep learning models with high success rates. As we will show later, point cloud data of these adversarial objects captured by 3D scanners may lead the deep learning models to arbitrary classification results controlled by the adversary.

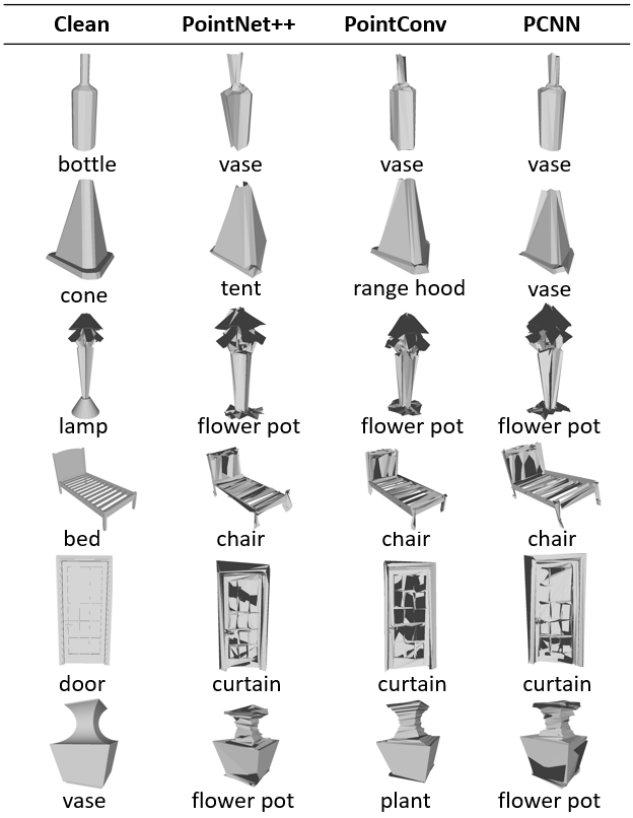

Fig. 1. Visualization of the 3D adversarial objects generated by our attack algorithm of each target class and model.

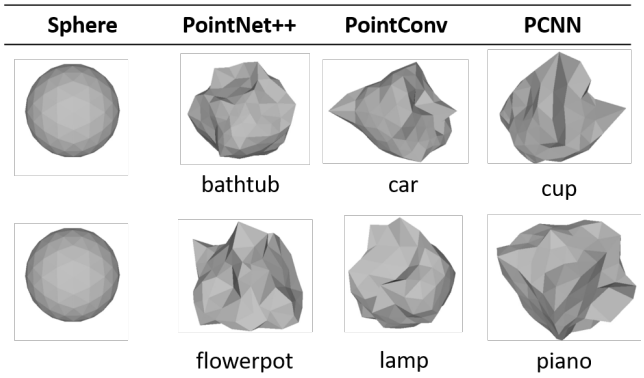

Fig. 2. Examples of the 3D reshaped sphere and the corresponding classification results regarding different target models.

D. Evaluation of 3D Printed Adversarial Objects

The adversarial meshes generated by the proposed methods are physically manufactured by 3D printers, and then be re-scanned by a 3D scanner. The captured point cloud are then used as inputs for the target models to verify the effectiveness of our attacks. In total, we print 10 objects with 5 reshaped objects and 5 altered objects. Among them, 9 of them receive desired classification results, showing that our methods are effective in the real-world scenario. Some examples of the printed adversarial objects with the attack type, target model and attack goal are shown in Figure 3. The scanning environment including the Einscan-SE scanner with the data collecting laptop is set as shown in Figure 4.

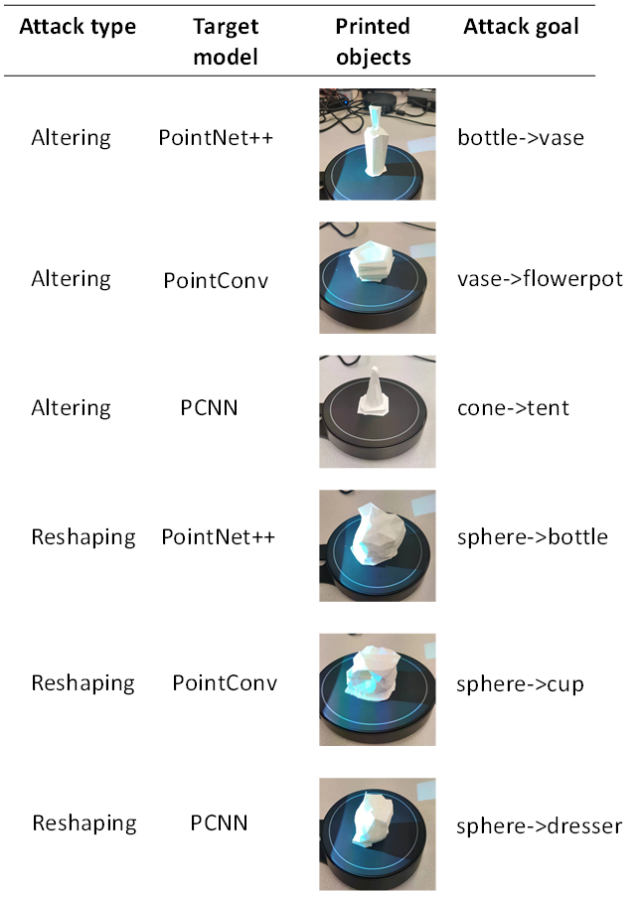

Fig. 3. Physical printed adversarial objects with their target models and attack goals.

TABLE II
MIS-CLASSIFICATION RATE OF OUR RESHAPING ATTACK USING STANDARD SPHERE.

Object	Models	Mis-classification Rate
sphere	PointNet++	36/40 (90.00%)
sphere	PointConv	35/40 (87.5%)
sphere	PCNN	37/40 (92.5%)

E. Comparison with Existing Works

The comparison of our work with existing works are shown in Table III and Table IV. In Table III the different settings of our attack and existing works are illustrated. Comparing to previous attacks, our method is designed for a more practical setting. All previous works are designed for white-box attacks against single type of deep learning models while our attacks can be launched in the black-box setting against multiple types of deep learning schemes. From Table IV, we can observe that our attack encounter slight success rate reduction comparing to some existing works in the simulation when only point cloud data is considered. We believe that this