Escaping Plato's Cave using Adversarial Training: 3D Shape From Unstructured 2D Image Collections

Philipp Henzler
p.henzler@cs.ucl.ac.uk

Niloy Mitra
n.mitra@cs.ucl.ac.uk

Tobias Ritschel
t.ritschel@ucl.ac.uk

University College London

Abstract

We develop PLATONICGAN *to discover 3D structure of an object class from an unstructured collection of 2D images. The key idea is to learn a deep neural network that generates 3D shapes that are never objectionable to a discriminator looking only at its 2D projections, i. e., renderings of the generated volumes. Using such a 2D instead of a 3D discriminator allows tapping into massive 2D image collections instead of relying on much smaller 3D data sets.*

To establish constraints between 2D image observation and their 3D interpretation we suggest a family of rendering layers *that are effectively back-propagatable. This family includes visual hull, absorption-only (akin to x-ray), and emission-absorption (that can resolve occlusion if multiple 3D points project to the same 2D pixel). These layers are studied both on synthetic and real data in an application to reconstruct of 3D shape from 2D images.*

1. Introduction

A key limitation to current generative models [33, 32, 10, 21, 29, 28] is the availability of suitable training data (e. g., 3D volumes, structured image sets, etc.) for supervision.

While methods exist to learn the 3D structure of classes of objects, they typically require 3D data as input. Regrettably, such 3D data is difficult to acquire, in particular for the "long tail" of exotic classes: ShapeNet might have `chair`, but it does not have `chanterelle`. Can we enable biologists to benefit from a generative model for such rarely observed classes?

Addressing this problem we suggest a method to learn 3D structure from 2D images only (Fig. 1). Reasoning about the 3D structure from 2D observations without assuming anything about their relation is challenging as illustrated from Plato's Allegory of the Cave [30]: *How can we hope to understand higher dimensions from only seeing projections?* If multiple views (maybe only two [36, 11]) of the same object are available, multi-view analysis without 3D supervision has been successful, but regrettably massive image collections do not come in this form but are now and will remain unstructured: they show random instances under random pose, uncalibrated lighting in unknown relations.

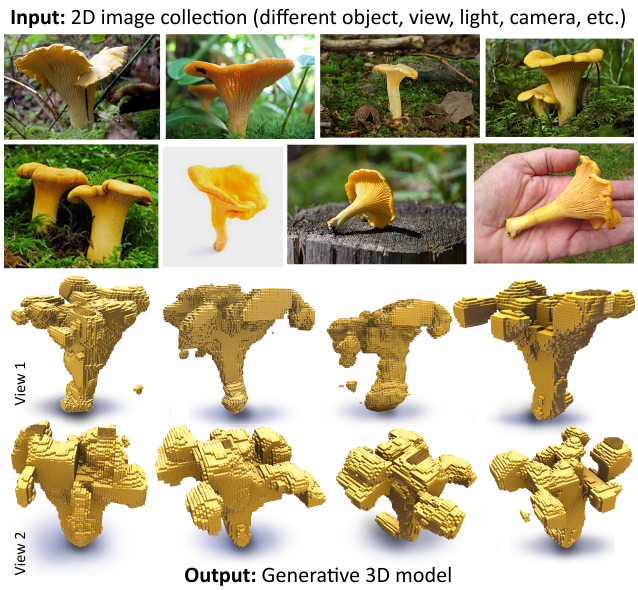

Figure 1. PlatonicGANs allow converting an unstructured collection of 2D images of a rare class (subset shown on top) into a generative 3D model (random samples below seen from two views).

Our first main contribution (Sec. 3) is to use adversarial training of a 3D generator that makes use of a discriminator that operates exclusively on widely available unstructured collections of 2D images, which we call *platonic discriminator*. Here, during training, the generator produces a 3D shape that is projected (rendered) to 2D and presented to a 2D discriminator. Making a connection between the 3D generator and the 2D discriminator, our second key contribution, is enabled by a family of *rendering layers* that can account for occlusion and color (Sec. 4). These layers do not have and do not need any learnable parameters and are efficient to back-propagate [24]. From these two key blocks

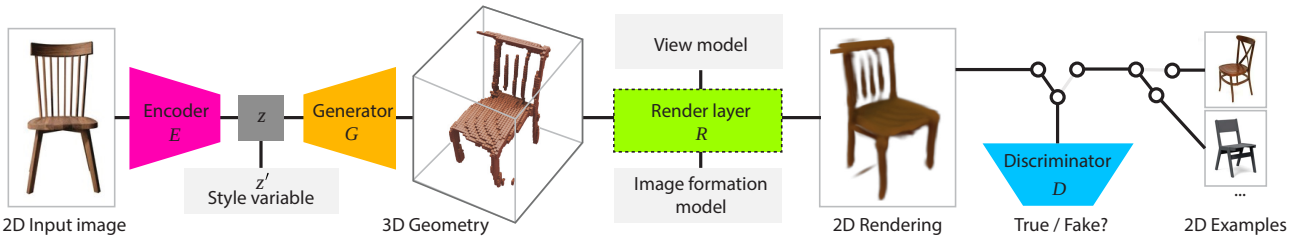

Figure 2. Overview of our method: We start from a 2D image which is encoded using an encoder E into a latent code $\mathbf{z}$ that is augmented by a style variable $\mathbf{z}'$ and fed into a generator G to produce a 3D volume. This 3D volume is inserted into a render layer R that maps it back to a 2D image which is presented to a critic D. The render layer is controlled by an image formation model (e. g., VH, AO, EA) and a model to sample views. This critic is trained on both such rendered imagery and on samples from an unstructured 3D image collection, i. e., images of the same class of objects, but never the same instance, view or lighting and with no assumptions about these or their relation.

we construct a system that learns the 3D shapes of common classes such as chairs and cars, but also exotic ones from unstructured 2D image collections. We demonstrate 3D reconstruction from a single 2D image as a key application (Sec. 5).

2. Related Work

Several papers suggest (adversarial) learning using 3D voxel representations [33, 32, 10, 21, 29, 28, 31, 35, 27, 17] or point cloud input [1, 9]. A simplified design for such a network is seen in Fig. 3, c, adapted to our task: an encoder generates a latent code that is fed into a generator to produce a 3D representation (i. e., a voxel grid). A 3D discriminator now analyzes samples both from the generator and from the ground truth distribution. Note that this procedure requires 3D supervision, i. e., is limited by the type and size of the 3D data set such as [4].

Girdhar et al. [10] work on a joint embedding of 3D voxels and 2D images, but still require 3D voxeliations as input. Fan et al. [8] produce points from 2D images, but similarly with 3D data as training input. Cho et al.'s recursive design takes multiple images as input [5] while also being trained on 3D data. Kar et al. [14] propose a simple "unprojection" network component to establish a relation between 2D pixels and 3D voxels but without resolving occlusion and again with 3D supervision.

Cashman and Fitzgibbon [3] have extracted template-based shape spaces from collections of 2D images. Similarly, Carreira et al. [2] use correspondence to templates across segmentation-labeled 2D image data sets to reconstruct 3D shapes.

Closer to our approach is Rezende et al. [22] that also learn 3D representations from 2D images. However, they make use of a partially differentiable renderer [19] that is limited to surface orientation and shading, while our formulation can resolve both occlusion from the camera and appearance. Also, their representation of the 3D volume is a latent one, that is, it has no immediate physical interpretation that is required in practice, e. g., for measurements, to run simulations such as renderings or 3D printing. This choice of having a deep representation of the 3D world is shared by Eslami et al. [7]. Tulsiani et al. [26] use a special case of our rendering layer (our ρ_{VH}) in a setting where multiple 2D images with a known camera transform for each are available at learning time. We take it a step further and use a GAN to work with unstructured single images at training time. In particular we do not even know the camera pose relative to the object in the image. Finally, our image formation goes beyond visual hulls accounting for color and occlusion.

While early suggestions how to extend differentiable renderers to polygonal meshes exist, they are limited to deformation of a pre-defined template [16]. We work with voxels, that can express arbitrary topology, e. g., we can generate airplanes with drastically different layout, which are not a mere deformation of a base shape. Combining our approach with sparse voxelizations [23] would allow to reproduce even finer details.

Similarly, inter-view constraints can be used to learn depth maps [36, 11] using reprojection constraints: If the depth label is correct, reprojecting one image into the other view has to produce the other image. Our method does not learn a single depth map but a full voxel grid and allows principled handling of occlusions.

A generalization from visual hull maps to full 3D scenes is discussed by Yan et al. [34], illustrated in Fig. 3, b: Instead of a 3D loss, they employ a simple projection along major axis allowing to use a 2D loss. However, multiple 2D images of the same object are required. In practice this is achieved by rendering the 3D shape into 2D images from multiple views. This makes two assumptions: We have multiple images in a *known* relation and available reference appearance (i. e., light, materials). Our approach relaxes those two requirements: we use a discriminator that can work on arbitrary projections and arbitrary natural input images, without known refehorence.

3. 3D Shape From 2D Image Collections

Here, we discuss our PLATONICGAN in more detail, making use of our rendering layers to be introduced in Sec. 4.

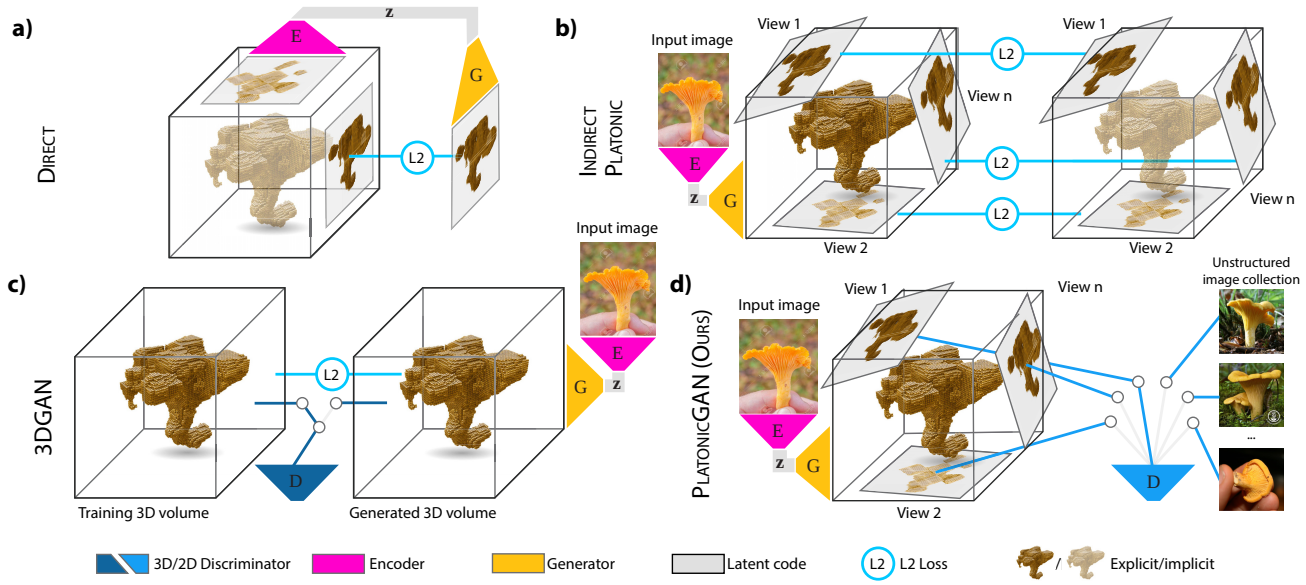

Figure 3. The evolution of generative 3D modeling, starting from prediction of novel views from an image ("Direct"), over indirect methods that have a loss on multiple views of the same scene ("Indirect") [34] to 3D GANs ("3D GAN") [32] and finally platonic approach ("Our").

Common GAN Our method is a classic (generative) adversarial design [12] with two main differences: The 2D discriminator D operates in a different space (2D) than the 3D generator G (3D) and the two are linked by a hard-wired projection operator (rendering layer, Sec. 4).

Let us recall the classic adversarial learning of 3D shapes [32], which is a min-max game

$$\min_{\Psi} \max_{\Theta} \; c_{\mathrm{Dat}}(\Psi) + c_{\mathrm{Gen}'}(\Theta) \tag{1}$$

between the data and the generator cost c_{Dat} and c_{Gen}.

The data cost is

$$c_{\mathrm{Dat}}(\Psi) = \mathrm{E}_{p_{\mathrm{Data}}(\mathbf{x})}[\log(1 - D_{\Psi}(\mathbf{x}))], \tag{2}$$

where D_{Ψ} is the discriminator with learned parameters Ψ which is presented with samples $\mathbf{x}$ from the distribution of real 3D shapes $\mathbf{x} \sim p_{\mathrm{Data}}$. Here E_p denotes the expected value of the distribution p.

The generation cost is

$$c_{\mathrm{Gen}'}(\Theta) = \mathrm{E}_{p_{\mathrm{Gen}}(\mathbf{z})}[\log(D_{\Psi}(G_{\Theta}(\mathbf{z}))] \tag{3}$$

where G_{Θ} is the generator with parameters Θ that maps the latent code $\mathbf{z} \sim p_{\mathrm{Gen}}$ to the data domain.

Platonic GANs extend the generator cost to

$$c_{\mathrm{Gen}}(\Theta) = \mathrm{E}_{p_{\mathrm{Gen}}(\mathbf{z})} \mathrm{E}_{p_{\mathrm{View}}(\omega)}[\log(D_{\Psi}(R(\omega, G_{\Theta}(\mathbf{z})))] \tag{4}$$

which projects the generator result $G_{\Theta}(\mathbf{z})$ from 3D to 2D along sampled view direction ω. Different implementations of R are discussed in Sec. 4.

While many parametrizations for views are possible, we here choose an orthographic camera that looks at the origin, with upright orientation from an Euclidean position $\omega \in \mathbb{R}^3$. $\mathrm{E}_{p_{\mathrm{View}}(\omega)}$ is the expected value across the distributions $\omega \sim p_{\mathrm{View}}$ of views.

Full Platonic GAN Additionally, we make use of three recent ideas orthogonal to our platonic concept, resulting in

$$\min_{\Psi} \max_{\Theta,\Phi} \; c_{\mathrm{Dat}}(\Psi) + c_{\mathrm{Gen}}(\Theta, \Phi) + c_{\mathrm{KL}}(\Phi) + c_{\mathrm{Rec}}(\Theta, \Phi), \tag{5}$$

where c_{Gen} includes an encoding step, c_{KL} enforces a normal distribution of latent codes and c_{Rec} encourgaes the encoded-generated-and-projected result to be similar to the encoder input. We detail each of these three steps in the following paragraphs:

Generator First, the final generator G_{Θ} does not directly work on a latent codes $\mathbf{z}$, but allows for an encoder E_{Ψ} with parameters Ψ that maps an input 2D image or 3D volume $\mathbf{y}$ to a latent code: $\mathbf{z}$

$$c_{\mathrm{Gen}}(\Theta, \Phi) = \mathrm{E}_{p_{\mathrm{Dat}}(\mathbf{x})}[\log(D_{\Psi}(G_{\Theta}(E_{\Phi}(\mathbf{x}))))]. \tag{6}$$

Using identity instead of a complex mapping E addresses a (random) generation task, while using images or volumes as input is a solution to reconstruction or filtering problems, respectively. Note, that the expected value for the generator is now across 2D images $\mathbf{x} \sim p_{\mathrm{Dat}}$ and not across the latent codes $\mathbf{z} \sim p_{\mathrm{Gen}}$ anymore.

Reconstruction Third, we encourage the encoder E and generator G to reproduce the input in the $\mathcal{L}_2$ sense, if the view ω is the input view, by convention $\omega = (0, 0, 1)$, so

$$c_{\text{Rec}}(\Theta, \Phi) = \mathrm{E}_{p_{\text{Dat}}} \|\mathbf{y} - R((0, 0, 1), G(E(\mathbf{x})))\|_2^2. \quad (7)$$

While this step is optional, as the original view as well as other views similar to it, is already contained in the generator term, we found it useful to give direct additional weighting to this special view. It adds stability to the optimization as it is easy to find an initial solution that matches this 2D cost before refining the 3D structure.

3.1. Optimization

Two properties enable optimizing for a Platonic GAN: First, maximizing the expected value across the distribution of views p_{View} and second, back-propagation through the projection operator R.
We extend the classic GAN optimization procedure to become Alg. 1 using $\ell(a, b) = b \log a + (1 - b) \log(1 - a)$ as the binary cross entropy loss. We use the regularization from Mescheder et al. [20] for training and increase variation by mini-batch standard deviation [15].

Algorithm 1 Platonic GAN Update Step

1: $\mathbf{x} \leftarrow \text{SAMPLEBATCH}(p_{\text{Dat}})$
2: $\Psi \leftarrow \text{MINIMIZE}(\ell(D_\Psi(\mathbf{x}), 1))$
3: $\mathbf{z} \leftarrow E_\Phi(\text{SAMPLEBATCH}(p_{\text{Dat}}))$
4: **for** n_{v} **do**
5: $\quad \omega \leftarrow \text{SAMPLEVIEWS}(p_{\text{View}})$
6: $\quad \Psi \leftarrow \text{MINIMIZE}(\ell(D_\Psi(R(\omega, G_\Psi(\mathbf{z}), 0)))$
7: $\quad \Theta, \Phi \leftarrow \text{MAXIMIZE}(\ell(D_\Psi(R(\omega, G_\Psi(\mathbf{z}), 1)))$
8: **end for**

View sampling Different options exist to sample the views in the `for` loop in Alg. 1, even if the the distribution p_{View} is uniform. A simple solution is to sample one view, entirely randomly, in every classic GAN update step. While this works, it introduces a lot of variance into the estimate of the gradient which ideally was over all views.

We therefore sample multiple view n_{v} for every update. This also is economical, as the result of $G_\Theta(E_\Phi(\mathbf{x}))$, a 3D volume, is costly to produce once, but simple to render multiple times.

A more refined strategy keeps those views fixed across multiple (n_{u}) updates. This allows the generator to get a set of certain views right, before moving to new ones.

Finally, we suggest an "umbrella" strategy. The intuition is, that generating a 3D volume from one view is simple: just extrude the contour in 3D; this would not look objectionable to any critic with limited view variation. To resolve errors from other views, it might be simplest to start fixing errors from views similar to a view that is already right. Incrementally, views become more varied, like opening an umbrella. This is achieved by choosing

$$\omega = (\sqrt{1 - u^2} \cos v, \sqrt{1 - u^2} \sin v, u)^\mathsf{T} \quad (8)$$
$$\text{with} \quad u = U(1 - min(2, c \cdot i), 1)$$
$$\text{and} \quad v = U(0, 2\pi),$$

where $U(a, b)$ returns a uniform random number from $[a, b]$, $c = .00005$ is a constant to control the opening speed and i the learning iteration.

Projection While many projection operators R between different dimensions could be conceived, we focus on the case of a 3D generator on a regular voxel grid and a 2D discriminator on a regular image in the next section 4. Methods to scale dense regular grids to sparse ones, exist [25] and should be applicable to our approach as well. Consequently $R(\omega, \mathbf{x}) \in \mathbb{R}^{3 \times n_{\text{h}}} \times \text{SO}^3 \to \mathbb{R}^{2 \times n_{\text{l}}}$ is a mapping from a three-dimensional voxel grid with n_{h} channels to a 2D image with n_{l} channels that depend on direction ω.

We further define $R(\omega, \mathbf{x}) := \rho(\mathsf{T}(\omega)\mathbf{x})$ into a linear transformation $\mathsf{T}(\omega)$ that depends on the view and an image formation function $\rho(\mathbf{x})$ that does not, i. e., is view-independent. The transformation is shared by all implementation of the rendering layer, so we will only discuss the key differences of ρ in the following and assume the rotation to have already been applied. Note that a rotation and a linear resampling is back-propagatable and provided, e. g., as `torch.nn.functional.grid_sample`. While we work in orthographic space, T could also be constructed to become perspective.

4. Rendering Layers

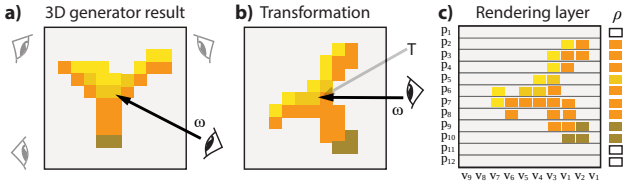

Figure 4. Rendering layers (Please see text).

Rendering layers (Fig. 4) map 3D information to 2D images so they can be presented to a discriminator. As explained, we have already rotated the 3D volume (Fig. 4, a) into camera space from view direction ω (Fig. 4, b), such that the pixel value p is to be computed from all voxel values v_i and only those (Fig. 4, c). Consequently a rendering layer is described by how it maps sequence of n_{z} voxels to a pixel value $r(\mathbf{v}) \in \mathbb{R}^{n_{\text{z}} \times n_{\text{g}}} \to \mathbb{R}^{n_{\text{c}}}$. Composing the full image ρ just amounts to executing ρ for every pixel p resp. all voxel $\mathbf{v} = v_1, \ldots, v_{n_{\text{z}}}$ at that pixel.

Note, that the rendering layer does not have any learnable parameters. It just serves as an additional constraint that allows the 3D generator to change in respect to loss observed in 2D. We will now discuss several variants of ρ, implementing different forms of volume rendering [6].

Visual hull (VH) Visual hull [18] is the simplest variant. It converts binary density voxels into binary opacity images, so $n_\mathrm{g} = n_\mathrm{c} = 1$. A voxel value of 0 means empty space and a value of 1 means occupied, i. e., $v_i \in \{0, 1\}$. Output again is a binary value indicating if any voxel blocked the ray. It simply states

$$\rho_{\mathrm{VH}}(\mathbf{v}) = 1 - \prod_i (1 - v_i).$$

Note that the product operator is both backprop and efficiently computable on a GPU using a parallel scan. We can apply this to learning 3D structure from binary 2D data such as segmented 2D images.

Absorption-only (AO) The absorption-only model is the gradual variant of visual hull. This allows for "softer" attenuation of rays. The same definition

$$\rho_{\mathrm{AO}}(\mathbf{v}) = 1 - \prod_i (1 - v_i),$$

can be applied, but if v_i are fractional the result is similar to an x-ray, i. e., $v_i \in [0, 1]$. This image formation allows learning from x-rays or other transparent 2D images. Typical, both are mono, but a colored method (e. g., x-ray at different wavelength or RGB images of colored transparent objects) could technically be done.

Emission-absorption (EA) Emission-absorption allows the voxels not only to absorb light coming towards the observer but also to emit new light at any position. This interplay of emission and absorption can model occlusion, which we will see is useful to make 3D sense of a 3D world. Fig. 4 uses emission-absorption with high absorption, effectively realizing an opaque surface with visibility.

A typical choice is to have the absorption v_a monochromatic and the emission v_e chromatic, so the voxels carry $n_\mathrm{g} = 4$ channels. Consequently, an output image has $n_\mathrm{c} = 3$ color channels.

The complete emission-absorption equation is

$$\rho_{\mathrm{EA}}(\mathbf{v}) = \sum_{i=1}^{n_z} \underbrace{\left(1 - \prod_{j=1}^{i} (1 - v_{\mathrm{a},j})\right)}_{\text{Transmission } t_i \text{ to voxel } i} v_{\mathrm{e},i}$$

While such equations are typically solved using ray-marching [6], they can be rewritten to become backprop in practice: First, we note that the transmission t_i from voxel i is one minus a product of one minus the density of all voxels before i. Such a cumulative product can be, similar to a sum, backproped and computed efficiently using parallel scans as well, e. g., using `torch.cumprod`. An numerical alternative, that performed similar in our experiemnts, is to work in the log domain and use `torch.cumsum`.

5. Evaluation

Our evaluation comprises of a quantitative (Sec. 5.3) and a qualitative analysis (Sec. 5.4) that compares different techniques (Sec. 5.2).

5.1. Data sets

We train on several synthetic and real data sets.

Synthetic Our synthetic data set comprises of 2D images of different modalities (opacity, RGB) that were rendered using ShapeNet [4]. We chose following classes: `airplane`, `car`, `chair`, `rifle` and `lamp` Each sample is rendered from a random view (50 per object), with random natural illumination, using the three image formation methods (VH, AO, EA) we suggest, producing three 2D images. No 3D information is used as training data in our approach. We use volumetric projective texturing to propagate the appearance information from thin 3D surface crust as defined by ShapeNet's textures into the 3D voxelization.

Furthermore, we also train on a synthetic x-ray dataset that consists of mammalia skulls. We used a subset (monkey skulls only) of that data set. [13]

Real We use two rare classes: `chanterelle` (60 images) and `tree` (37 images) (not strictly rare, but difficult to 3D-model). These images are RGBA, masked, on white background.

5.2. Techniques

We compare different alternative methods (Fig. 3) against our method. A key property of our approach is not to require access to the 3D volume. We investigate three variants where two require more supervision than our method.

Direct First, we train a DIRECT method, that avoids reconstructing a 3D volume altogether and directly produces a 2D image from the input image using a simple Encoder-Decoder network (Fig. 3, a). While this does not produce a 3D representation and does not allow for applications such as using it in a modeling package or 3D printing. It indicates how our effort to construct a non-deep 3D representation can pay off, even if it is just to produce a novel-view 2D image.

Indirect Second, we investigate a INDIRECT [34] method, that does not have access to the 3D volume but instead to 2D multiple images of that 3D volume in a known spatial relation (Fig. 3, b). Note, that this is a stronger requirement to our approach that does not require any structure in the adversarial examples: geometry, view, light, all change, while in this method only the view changes in a prescribed way.

3DGAN Third, we study a classic 3DGAN [32], that has access to the 3D volumes at run-time (Fig. 3, c). The architecture is the same as ours, just that the discriminator operates directly in 3D instead of 2D.

Ours We investigate variants of our approach mainly in the way the novel views are generated. We explore two axes: The number of views n_{v} and the update protocol that either updates every n_{u} iterations or applies the umbrella strategy.

Please also note that for all, but our rare classes (which only we can process), our approach can only ever hope to perform as good as those methods on our data. In other words, the chosen variants are upper bound to our method as they require access to more structured data.

5.3. Quantitative

Methods We first report RMSE on 3D voxels. Regrettably, this does not capture well the effect on 2D images. Thus, we render both the reconstructed and the reference volume from the same 5 random views and compare their images using SSIM. For this re-rendering, we further employ four different rendering methods: the original (i. e., ρ) image formation (IF), volume rendering (VOL), iso-surface rendering with an iso-val of .1 (ISO) and a voxel rendering (VOX), all under random natural illumination.

Results Tbl. 1 summarizes our main results for the prototypical class `airplane`. We see that overall, our 2D supervised method produces competitive 2D SSIM and 3D MSE as the 3D supervised methods.

In terms of 3D MSE, not surprising, the 3DGAN wins, as it has this measure in its loss. Our method performs slightly better than half as good. Note that 3D MSE is receptive of small alignment errors that our critic does not, and also should not, consider. The DIRECT rows do not state 3D MSE, as it does produce 2D images only.

We see that our approach is close to the SSIM of the 3D-supervised methods, and for the case of voxel rendering even surpassing them. The DIRECT method can only be applied to the IF visualization, as since it does not produce 3D volumes that could be re-rendered. We see that direct regression always performs worse than ours in SSIM. Typically, its results are blurry, while ours are sharp. Overall, the 3DGAN performs better than the INDIRECT method, but often their differences are of similar magnitude than ours to both of them.

Table 1. Performance of different methods **(rows)** on different metrics **(columns)** for the class `airplane`.

Method	Image Form. (IF)	Reconstruction Task				
		3D MSE	2D SSIM			
			IF	VOL	ISO	VOX
OUR	 IF=VH	.158	.872	.932	.920	.926
DIRECT		—	.560	—	—	—
INDIRECT		.130	**.880**	**.938**	**.928**	**.933**
3DGAN		**.111**	.833	.924	.921	.926
OUR	 IF=AO	.113	**.950**	**.935**	**.927**	**.932**
DIRECT		—	.641	—	—	—
INDIRECT		.115	.945	.932	.921	.927
3DGAN		**.108**	.934	.922	.919	.923
OUR	 IF=EA	.220	.837	.837	.757	.766
DIRECT		—	.701	.701	—	—
INDIRECT		.109	**.922**	**.922**	**.902**	**.907**
3DGAN		**.107**	.920	.920	.883	.893

Table 2. Reconstruction performance of our method for different image formation models **(columns)** on different classes **(rows)**.

Class	VH			AO			EA		
	VOL	ISO	VOX	VOL	ISO	VOX	VOL	ISO	VOX
`plane`	.932	.920	.926	.935	.927	.932	.837	.757	.766
`rifle`	.946	.941	.945	.949	.944	.947	.899	.777	.803
`chair`	.860	.848	.854	.862	.850	.857	.802	.611	.632
`car`	.841	.846	.851	.844	.846	.850	.800	.731	.743
`lamp`	.920	.915	.920	.926	.914	.920	.883	.790	.803

Concerning the image formation models, we see that the absolute SSIM values are best for AO, which is expected: VH asks for scalar density but has only a binary image; AO provides internal structures but only needs to produce scalar density; EA is hardest, as it needs to resolve both density and color. Nonetheless the differences between us and competitors are similar across the image formation models.

In Tbl. 2 we look into the performance across different classes. We see that our method produces acceptable SSIM across the board (compare to the 3DGAN / INDIRECT competitor SSIMs for the `airplane` class in Tbl. 1). `rifle` seems to work best: the approach learns quickly from 2D that a gun has an outer 3D shape that is a revolute structure. `chair` remains a different class likely due to its high intra-class variation.

Figure 5. Visual results for the reconstruction for three classes (`airplane`, `chair`, `rifle`) from multiple views.

5.4. Qualitative

Fig. 5 shows typical results for the reconstruction task. We see that our reconstruction can produce `airplane` 3D model representative of the input 2D image. Most importantly, these 3D models look plausible for multiple views, not only from the input one, as seen from the second view column. We also see that the model captures the relevant variation, ranging from jet fighters over smaller sports aircrafts to large transport types. `chair` is a particularly difficult class, but our results capture most variation in legs or thickness but small errors are present mostly in the form of sporadic activations in free space. For `gun`, the results turn out almost perfect, in agreement with the numbers reported before. In summary, we think the quality is comparable to supervised GANs, but 3D supervision.

6. Discussion

Which and how many views are needed? The key to PLATONICGAN is 3D-2D projection, but multiple strategies are conceivable to do so. Two parameter control its effect: the number of projections n_{v} and the protocol for their update, which is either every n_{u} times or following an umbrella. Tbl. 3 shows the error depending on different viewing protocols. We see, that the optimal combination is for a single view using the umbrella strategy.

Why not having a multi-view discriminator? It is tempting to suggest a discriminator that does not only look at a single image, but at multiple views at he same time to judge if the generator result is plausible holistically. But while we can generate "fake" images from multiple views, this is not possible, as p_{Data}, the set of "real" natural images does not come in such a form. As a key advantage, ours only expects unstructured data: online repositories hold images with unknown camera, 3D geometry or illumination.

Table 3. Performance in SSIM (more is better) for different view numbers n_{v} **(columns)** and update frequency n_{u} **(rows)**.

Protocol	$n_{\mathrm{v}}=1$			$n_{\mathrm{v}}=5$			$n_{\mathrm{v}}=10$		
	VOL	ISO	VOX	VOL	ISO	VOX	VOL	ISO	VOX
$n_{\mathrm{u}}=10^0$	.930	.915	.922	.929	.918	.924	.931	.916	.923
$n_{\mathrm{u}}=10^2$	.934	**.927**	.931	.932	.921	.927	.932	.921	.927
$n_{\mathrm{u}}=10^3$	.933	.925	.930	.930	.918	.924	.932	.922	.928
Umbrella	**.935**	**.927**	**.932**	.932	.921	.927	.932	.918	.926

Rare classes Results for rare classes are seen in Fig. 1 and Fig. 6. We see that our method produces plausible details from multiple view while respecting the input image, even in this difficult case. No metric can be applied to these data as no 3D volume is available to compare in 3D or re-project.

We also explored reconstructing skulls form real x-ray (i. e., the AO IF model) images [13] in Fig. 7. We find the method to recover both external and internal structures.

Supplemental Our supplemental materials show novel-view videos, more analysis and both data and network definitions will be made publicly available upon publication.

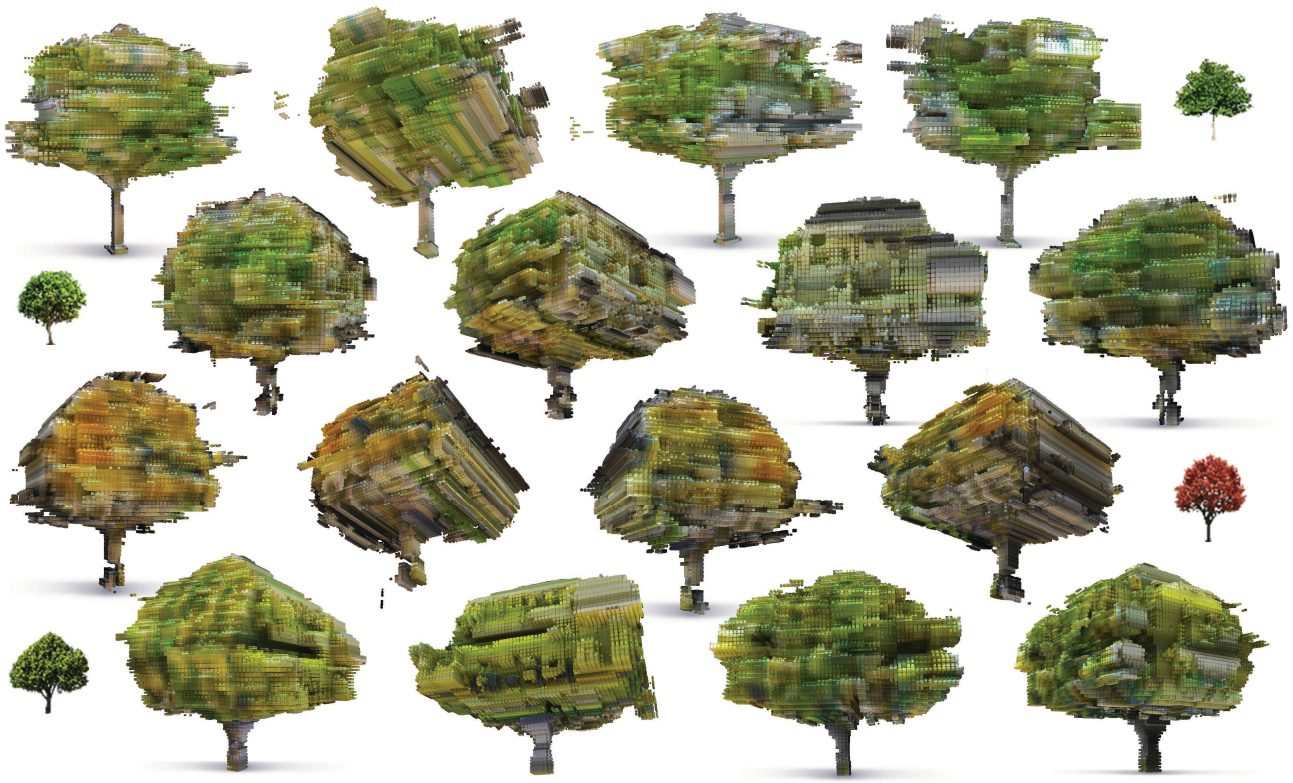

Figure 6. 3D Reconstruction of different trees using the EA IF from 2D images **(rows)**, seen from different views **(columns)**. The small images show the 2D images we reconstruct from. We see that PLATONICGAN has understood the 3D structure, including a distinctly colored stem, fractal geometry and structured leave textures. Please note, how the red color of the third exemplar is reproduced in 3D.

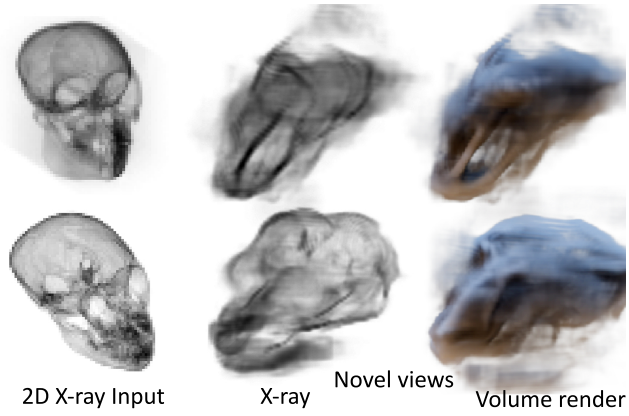

Figure 7. Platonic GANs trained on 2D x-rays (i. e., AO IF) of mammalian skulls **(left)**. The resulting 3D volumes can be rendered from novel views using x-ray **(middle)** and under novel views in different appearance, here, using image-based lighting **(right)**.

7. Conclusion

In this paper, we have presented PLATONICGAN, a new approach to learn 3D shape from unstructured collections of 2D images. The key to our "escape plan" is to train a 3D generator outside the cave that will fool a discriminator seeing projection inside the cave.

We have shown a family of rendering operators that can be GPU-efficiently back-propagated and account for occlusion and color. These support a range of input modalities, ranging from binary masks, over opacity maps to RGB images with transparency. Our 3D reconstruction application is build on top of this idea to capture varied and detailed 3D shapes, including color, from 2D images. Training is exclusively performed on 2D images, enabling massive 2D image collections to contribute to generating 3D shapes.

Future work could include shading that is related to gradients of density [6] into classic volume rendering. Furthermore, any sort of back-propagatable rendering operator ρ can be added. Devising such operators is a key future challenge. Other adversarial applications such as pure generation or filtering of 3D shapes seem worth exploring.

While we combine 2D observations with 3D interpretations, similar relations might exist in higher dimensions, between 3D observations and 4D (3D shapes in motion) but also in lower dimensions, such as for 1D row scanner in robotics or 2D slices of 3D data such as in tomography.

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. 2018. 2

[2] J. Carreira, S. Vicente, L. Agapito, and J. Batista. Lifting object detection datasets into 3d. *IEEE PAMI*, 38(7):1342–55, 2016. 2

[3] T. J. Cashman and A. W. Fitzgibbon. What shape are dolphins? building 3D morphable models from 2D images. *PAMI*, 35(1):232–44, 2013. 2

[4] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An information-rich 3D model repository. *arXiv:1512.03012*, 2015. 2, 5

[5] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *ECCV*, pages 628–44, 2016. 2

[6] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. In *Siggraph Computer Graphics*, volume 22, pages 65–74, 1988. 5, 8

[7] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, et al. Neural scene representation and rendering. *Science*, 360(6394):1204–10, 2018. 2

[8] H. Fan, H. Su, and L. Guibas. A point set generation network for 3D object reconstruction from a single image. *arXiv:1612.00603*, 2016. 2

[9] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, volume 2, page 6, 2017. 2

[10] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, pages 484–99, 2016. 1, 2

[11] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, pages 6602–6611, 2017. 1, 2

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–80, 2014. 3

[13] P. Henzler, V. Rasche, T. Ropinski, and T. Ritschel. Single-Image Tomography: 3D Volumes from 2D Cranial X-Rays. *Computer Graphics Forum (Proceedings of Eurographics 2018)*, 2018. 5, 7

[14] A. Kar, C. Häne, and J. Malik. Learning a multi-view stereo machine. In *NIPS*, pages 365–376, 2017. 2

[15] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *arXiv:1710.10196*, 2017. 4

[16] H. Kato, Y. Ushiku, and T. Harada. Neural 3D mesh renderer. In *CVPR*, pages 3907–16, 2018. 2

[17] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):29, 2013. 2

[18] A. Laurentini. The visual hull concept for silhouette-based image understanding. *AMI*, 16(2):150–62, 1994. 5

[19] M. M. Loper and M. J. Black. OpenDR: An approximate differentiable renderer. In *ECCV*, volume 8695, pages 154–69, 2014. 2

[20] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for GANs do actually converge? In *ICML*, pages 3478–87, 2018. 4

[21] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3D data. In *CVPR*, pages 5648–5656, 2016. 1, 2

[22] D. J. Rezende, S. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3D structure from images. In *NIPS*, pages 4996–5004, 2016. 2

[23] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017. 2

[24] D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988. 1

[25] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs. *arXiv:1703.09438*, 2017. 4

[26] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, 2017. 2

[27] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen. Shape completion enabled robotic grasping. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2442–2447. IEEE, 2017. 2

[28] H. Wang, J. Yang, W. Liang, and X. Tong. Deep single-view 3D object reconstruction with visual hull embedding. *arXiv:1809.03451*, 2018. 1, 2

[29] W. Wang, Q. Huang, S. You, C. Yang, and U. Neumann. Shape inpainting using 3d generative adversarial network and recurrent convolutional networks. *arXiv:1711.06375*, 2017. 1, 2

[30] E. H. Warmington, P. G. Rouse, and W. Rouse. *Great dialogues of Plato*. New American Library, 1956. 1

[31] J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum. MarrNet: 3D shape reconstruction via 2.5D sketches. In *NIPS*, pages 540–550, 2017. 2

[32] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *NIPS*, pages 82–90, 2016. 1, 2, 3, 6

[33] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D Shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–20, 2015. 1, 2

[34] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3D object reconstruction without 3D supervision. In *NIPS*, pages 1696–1704, 2016. 2, 3, 6

[35] B. Yang, H. Wen, S. Wang, R. Clark, A. Markham, and N. Trigoni. 3d object reconstruction from a single depth view with adversarial learning. *arXiv preprint arXiv:1708.07969*, 2017. 2

[36] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 1, 2