PointE: A System for Generating 3D Point Clouds from Complex Prompts

We refer to our system as **PointE**, since it generates **point** clouds **e**fficiently

Автор-докладчик: Анастасия Колчина
БПМИ191

Motivation

Как обычно принято генерить 3D облака точек:

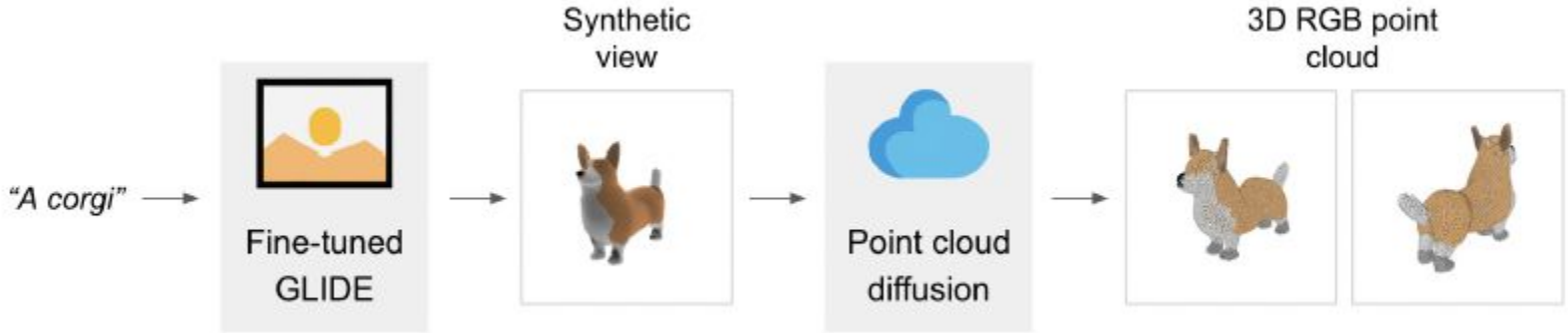
- с помощью GAN-ов по (text, 3D) или неразмеченным 3D данным - эффективно, но не работает на сложных промптах из-за нехватки больших 3D датасетов
- с помощью предобученных text-to-image моделей для оптимизации 3D представлений - осиливают сложные промпты, но рискуют сойтись в локальный минимум и нарисовать плохо

Решили совместить оба подхода через комбинацию text-to-image модели с image-to-3D, чтобы уметь справляться со сложными промптами эффективно.

Большинство методов работают по несколько часов на GPU, авторский метод же работает за пару минут на 1-ом GPU.

How?

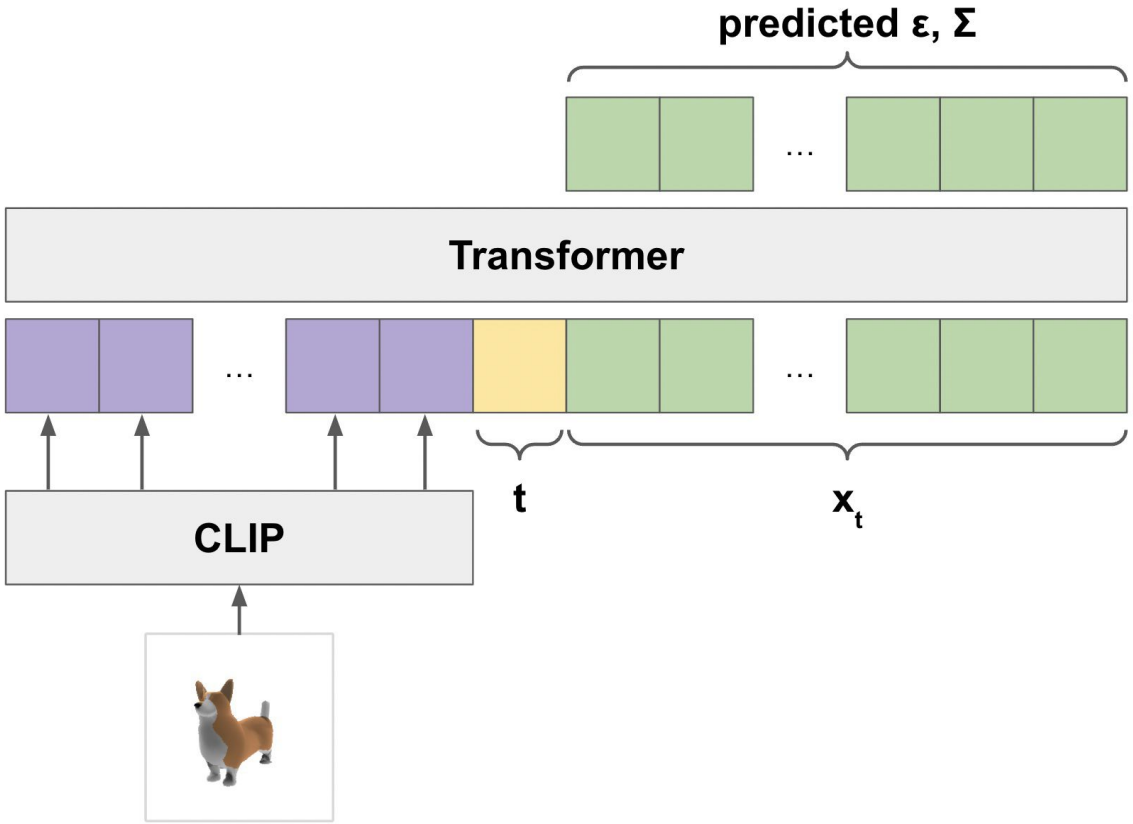
1. С помощью диффузионной модели GLIDE, зафантьюненной на 3D моделях из авторского датасета, генерим по текстовому промпту синтетическое изображение (text-to-image)
2. С помощью permutation invariant диффузионки генерим облако с низким разрешением по картинке (image-to-3D: 1024 точки)
3. С помощью еще более маленькой диффузионки по полученным облакам строим более качественные (4096 точек)
4. С помощью регрессионного подхода генерим сетку (mesh)



Architecture

1) text-to-image
2) image-to-3D
3) upsampling 3D объекта, полученного по картинке

Каждый этап занимает по времени пару секунд.



Dataset

Несколько миллионов 3D моделек.

Использовали Blender, чтобы привести их к одному формату: рендерим с 20 углов RGBAD картинки, затем для каждой нормализуем в bounding кубе, корректируем освещенность.

Затем каждый объект превращаем в цветное облако точек (4К точек) путем подсчета точки для каждого пикселя RGBAD картинки (то есть получается, что модельки строим не сразу по 3D сетке, а вот так, чтобы избежать проблем с форматом)

Чтобы уменьшить количество низкокачественных моделек, делаем SVD для каждой точки и оставляем те, у кого минимальное сингулярное значение меньше порога.

Также в процессе разбивки по бакетам для итогового датасета следили за сбалансированностью содержания плохих моделек.

View Synthesis GLIDE Model

Файнтьюним GLIDE на смеси 3D картинок из оригинального датасета (95%) и авторского (5%) на 100К итераций (т.е. несколько эпох на 3D датасете, но один и тот же 3D рендер дважды не смотрим).

На каждый текстовый промпт для 3D рендеров добавляем специальный токен, чтобы знать, что это для генерации 3D рендера, и сэмплируем всегда с этим токеном на тесте.

Point Cloud Diffusion

Облако точек - это тензор размера K x 6, где K - число точек, а вторая размерность состоит из координат (x, y, z) и цвета (R, G, B). Все координаты и цвета нормализованы в диапазоне [-1, 1]. Соответственно, с помощью диффузионки стартуем со случайного шума размера K x 6, постепенно разшумляя его.

Используем Transformer-based подход: предсказываем ϵ и $\sum$ по картинке, шаг t и зашумленное облако x_t. Каждую точку для входного контекста прогоняем через линейный слой размерности D, таким образом получая тензор K x D, а t прогоняем через маленькую MLP, чтобы получить еще один D-размерный вектор.

Затем прогоняем рендер через ViT-L/14 CLIP модель и берем эмбеддинги последнего слоя размера 256 x D', проецируем на тензор размера 256 x D и загоняем в Трансформер..

Финальный output имеет размер (K + 257) x D. Чтобы получить финальный ответ, берем последние K токенов и проецируем их, получая предсказанные ϵ и $\sum$.

Point Cloud Upsampler

Апсэмплим облако точек низкого разрешения (1К) до высокого (4К) с помощью небольшой upsampling модели.

Архитектура такая же, как у базовой модели, но для облаков низкого разрешения (1К) добавляем специальный токен. Чтобы избежать позиционных эмбеддингов, будем генерить 3К точек, добавлять их к низкоуровневым, и пропускать низкоуровневые (которые с токеном) через отдельный линейный слой для эмбеддингов.

Producing Meshes

Для построения сеток пробовали предобученные SAP-модели, но идея провалилась, т.к. у авторов генерится в облаках много выбросов.

Поэтому использовали регрессионный подход: с помощью регрессионной модели предсказываем по облаку для объекта размер его поля, затем применяем подгоняем его в куб и для каждой вершины полученной сетки ставим цвет наиболее близкой точки из оригинального облака.

Metrics

- **CLIP R-Precision** для оценки text-to-3D
- Вводятся новые метрики **P-IS** и **P-FID** (аналоги Inception Score и FID для облака точек). Для их построения используется модель PointNet++ для извлечения фичей и предсказания вероятностей классов для облаков точек.

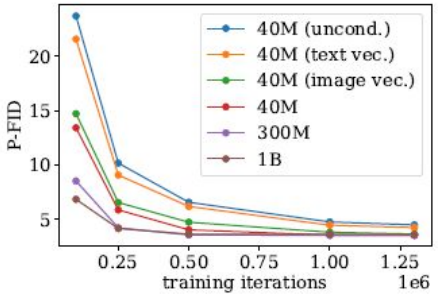
Ablation Study

Чтобы доказать важность image conditioning, авторы перебирали несколько базовых диффузионок (upsampler и conditioining картинки у всех одинаковые):

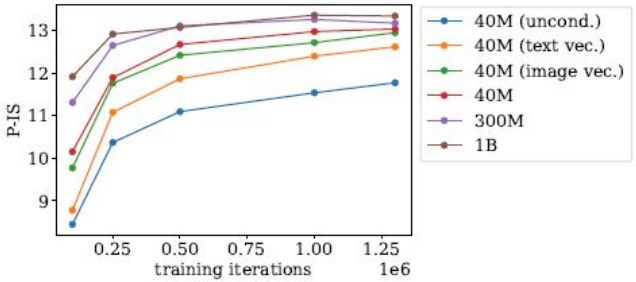
- **40M (uncond.):** маленькая модель без conditional информации
- **40M (text vec.):** conditional только по текстовым векторам, из которых затем получали эмбеддинг с помощью CLIP, который добавлялся как добавочный токен к контексту
- **40M (image vec.):** маленькая модель, conditional по эмбеддингам картинок из CLIP (тоже превращаем в единственный токен контекста)
- **40M:** маленькая full image conditioning модель
- **300M:** средняя
- **1B:** большая

Ablation Study. Итоги

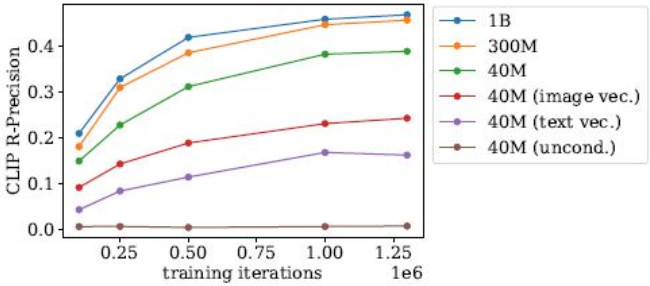
- Использование только text conditioning дает худший CLIP R-Precision
- Использование единственного эмбеддинга CLIP хуже, чем сетки (grid) эмбеддингов (теряем пространственную информацию о картинке)
- Скейлинг модели увеличивает скорость сходимости P-FID и увеличивает CLIP R-Precisioin.



(a) P-FID



(b) P-IS



(c) CLIP R-Precision

Comparison to Other Methods

Авторы утверждают, что худшие результаты их работы обусловлены тем, что:

- PointE не оптимизирует изображение с каждого угла, чтобы максимально соответствовать промпту
- для авторского метода облака точек должны быть предобработаны перед рендерингом - в общем, можно потерять информацию о самих точках из-за последующего построения сетки

Table 1. Comparison of Point·E to other 3D generative techniques as measured by CLIP R-Precision (with two different CLIP base models) on COCO evaluation prompts. *50 P100-minutes converted to V100-minutes using conversion rate $\frac{1}{3}$. †Assuming 2 V100 minutes = 1 A100 minute and 1 TPUv4-minute = 1 A100-minute. We report DreamFields results from Poole et al. (2022).

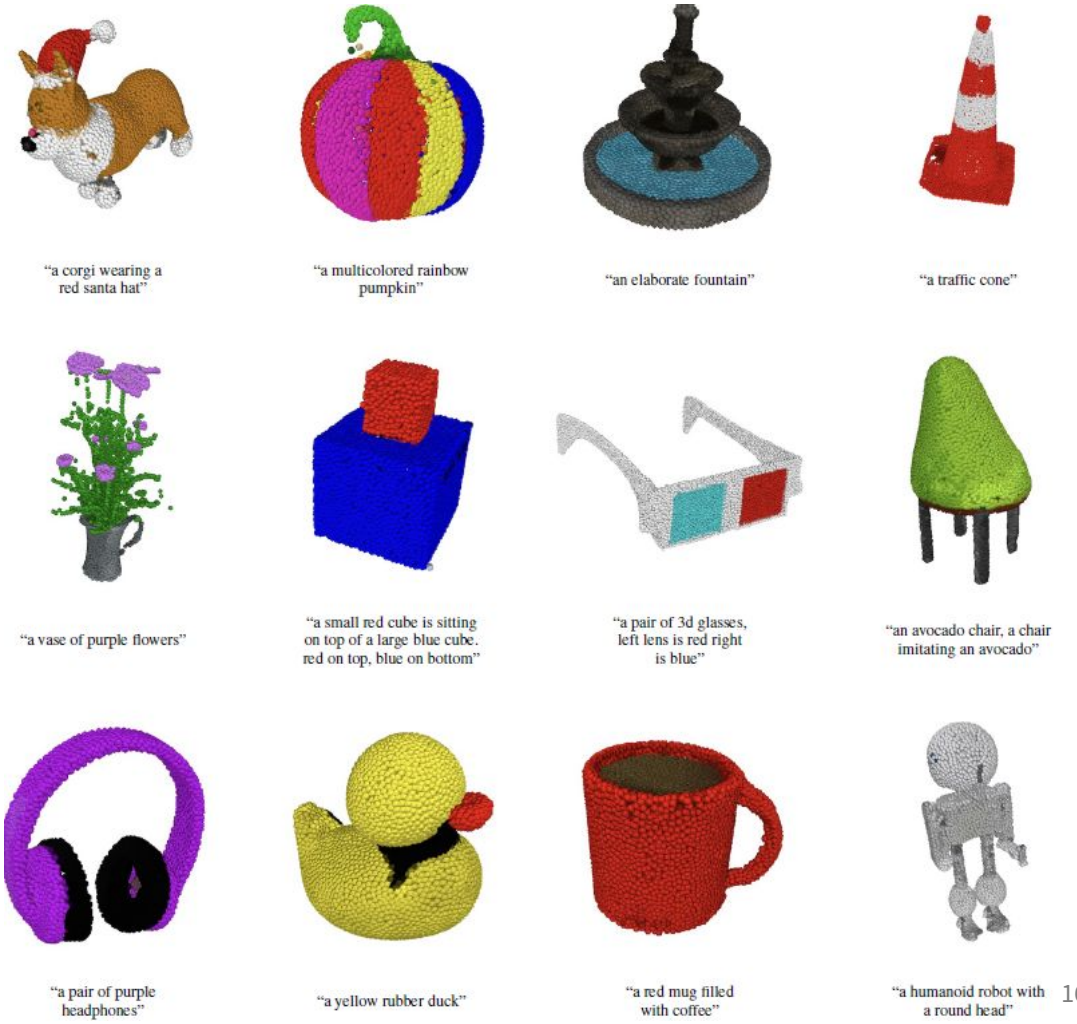
Method	ViT-B/32	ViT-L/14	Latency
DreamFields	78.6%	82.9%	∼ 200 V100-hr†
CLIP-Mesh	67.8%	74.5%	∼ 17 V100-min*
DreamFusion	75.1%	79.7%	∼ 12 V100-hr†
Point·E (40M, text-only)	15.4%	16.2%	16 V100-sec
Point·E (40M)	36.5%	38.8%	1.0 V100-min
Point·E (300M)	40.3%	45.6%	1.2 V100-min
Point·E (1B)	41.1%	46.8%	1.5 V100-min
Conditioning images	69.6%	86.6%	-

Future Usage & Prospects

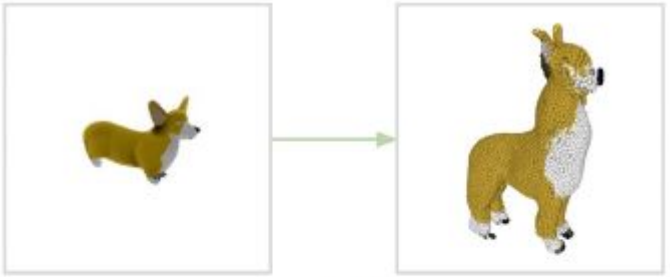
- Работает быстро - можно было бы насемплить кучу картинок и выбрать наиболее оптимальную за короткое время
- Работает хуже аналогов по метрике CLIP R-Precision
- Полученные облака точек низкого разрешения и не отражают точную форму и текстуру объекта
- Требуются синтетические рендеры - нужно подумать о создании 3D генераторов по настоящим изображениям
- Итого: это просто стартовая точка для дальнейших исследований

Pics

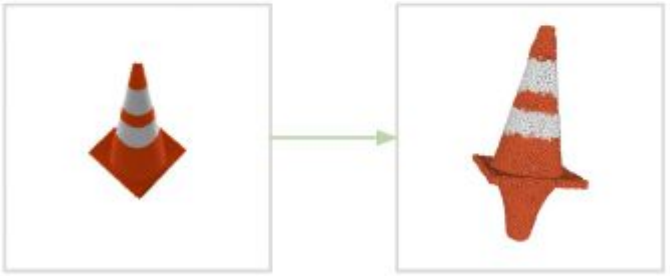
Selected point clouds generated by PointE using the given text prompts. For each prompt, we selected one point cloud out of eight samples.



Two common failure modes of our model.

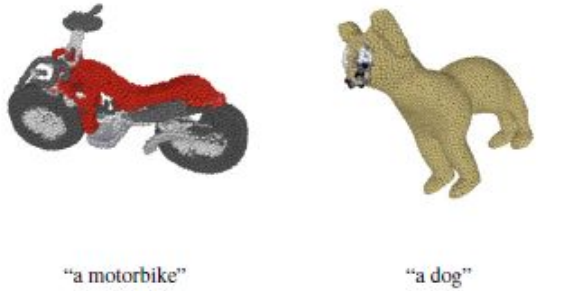


(a) Image to point cloud sample for the prompt “a very realistic 3D rendering of a corgi”.



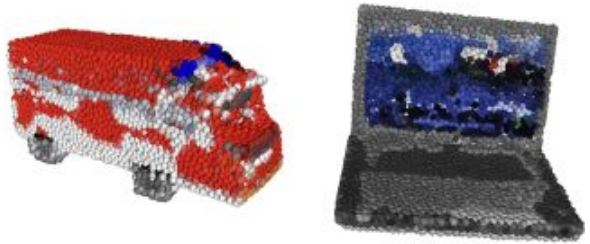
(b) Image to point cloud sample for the prompt “a traffic cone”.

Selected point clouds generated by our pure text-conditional 40M parameter point cloud diffusion model.



"a motorbike" "a dog"

"a desk lamp" "a guitar"



"an ambulance" "a laptop computer"

Examples of point clouds (left) and corresponding extracted meshes (right).