



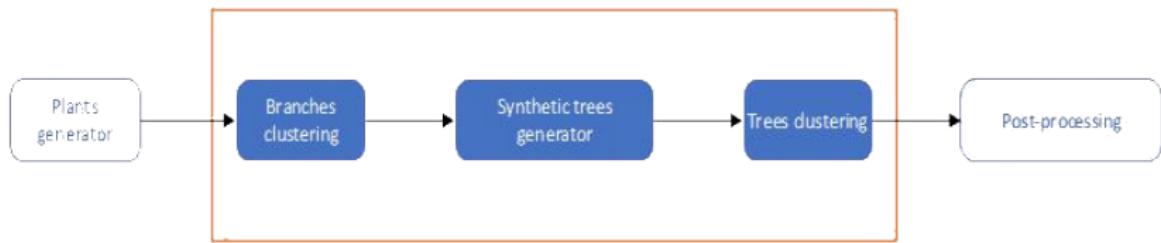
Approximate instancing method for plant ecosystems modeling

Альберт Гарифуллин,
Владимир Фролов,
Анастасия Хлюпина

Проблемы процедурной генерации

- Использование большого числа уникальных моделей требует много памяти
- Использование небольшого числа заранее подготовленных моделей не обеспечивает желаемого разнообразия
- Хорошие генераторы работают медленно
Как сохранить реалистичность и разнообразие процедурно генерируемых растений без необходимости хранить детализированные модели?

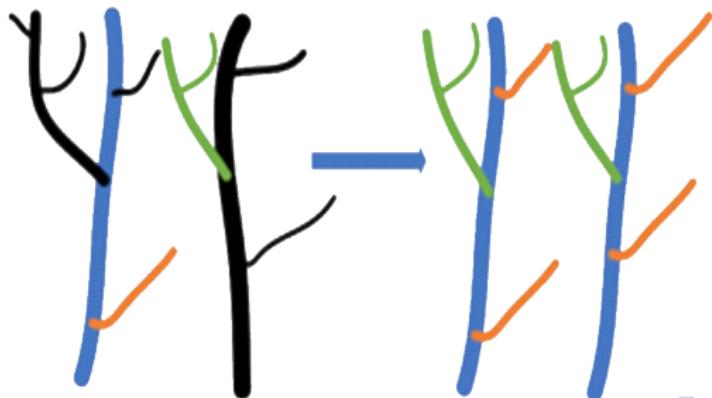
General pipeline



- Внешний генератор создает деревья в определенном формате
- Они трансформируются в компактное представление
- Результат может быть сконвертирован для хранения и использования - например, в GLTF-сцену

Approximate instancing

По множеству уникальных деревьев построить набор базовых структурных элементов, из которых, используя простые геометрические преобразования, можно получить деревья, внешне минимально отличающиеся от исходных.



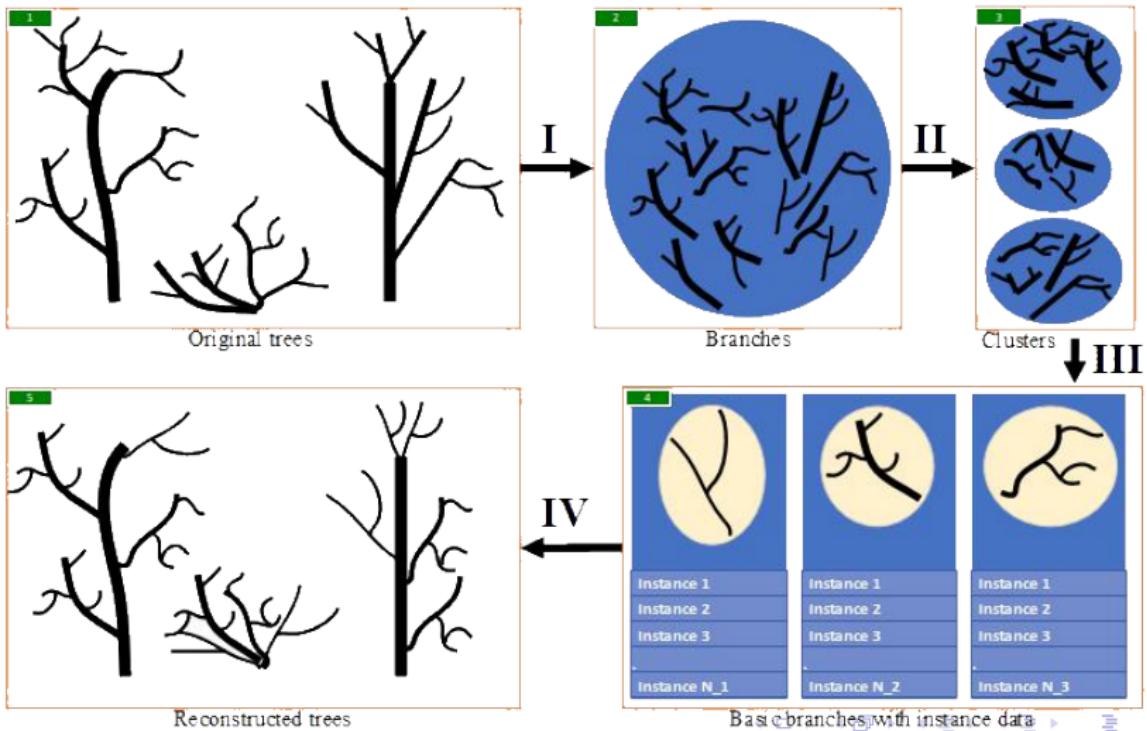
Кластеризация

Дерево представляется как структура, состоящая из ствола и веток, растущих из него.

Множества всех стволов и веток множества деревьев по отдельности проходят процедуру кластеризации - разделение на группы структурно похожих между собой элементов.

Все элементы одного кластера заменяются на instance типичного представителя

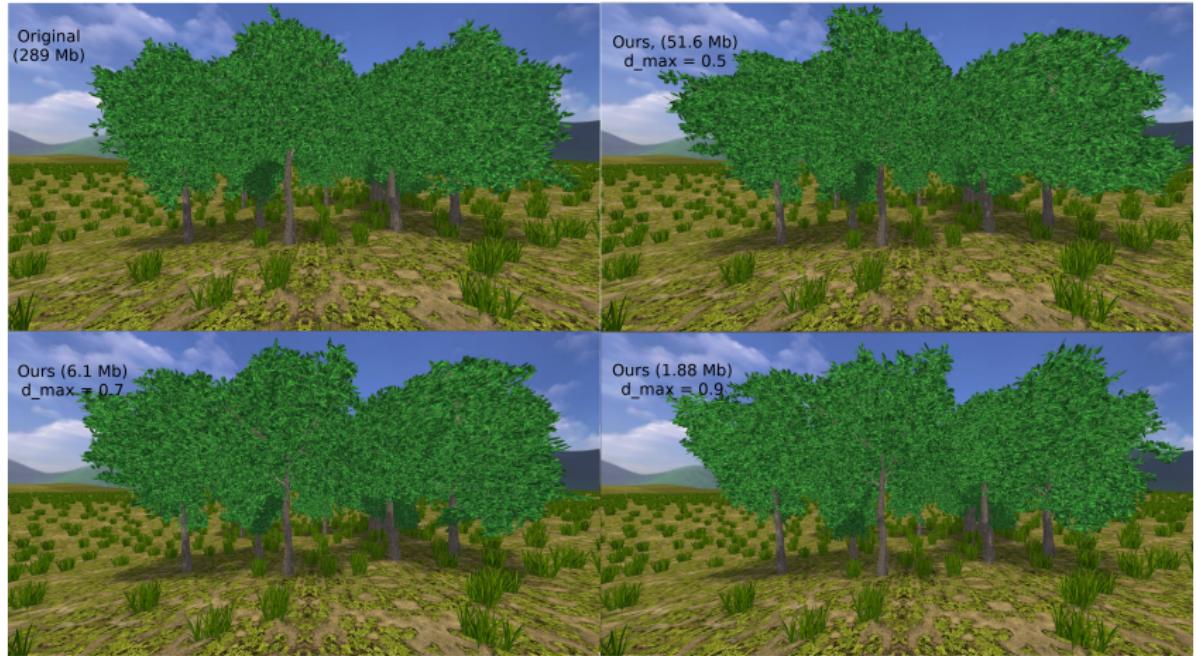
Кластеризация



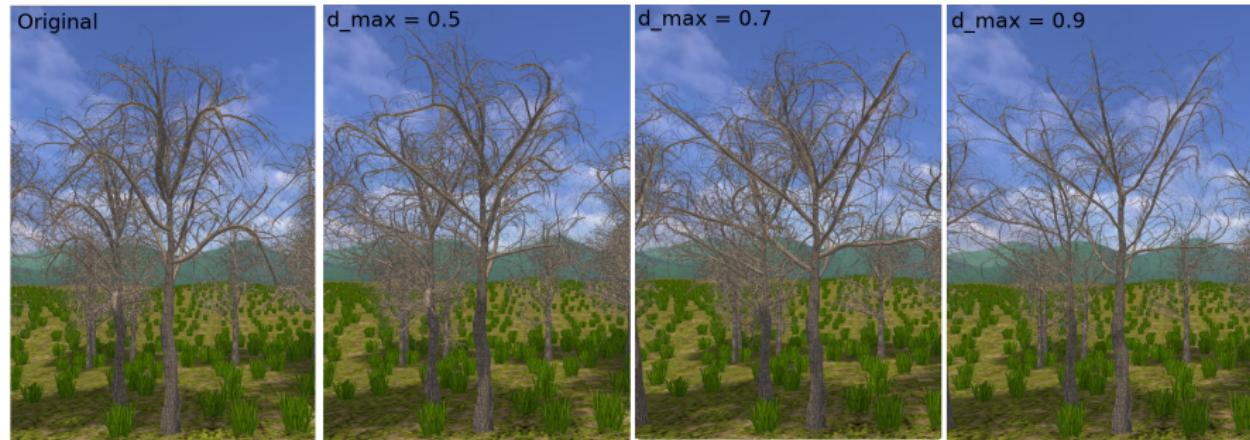
Функция дистанции

- Приведение веток в нормализованную форму
- Сравнение вокселизованных моделей
- Поиск структурных соответствий - попарное соотнесение узлов друг с другом

Результат

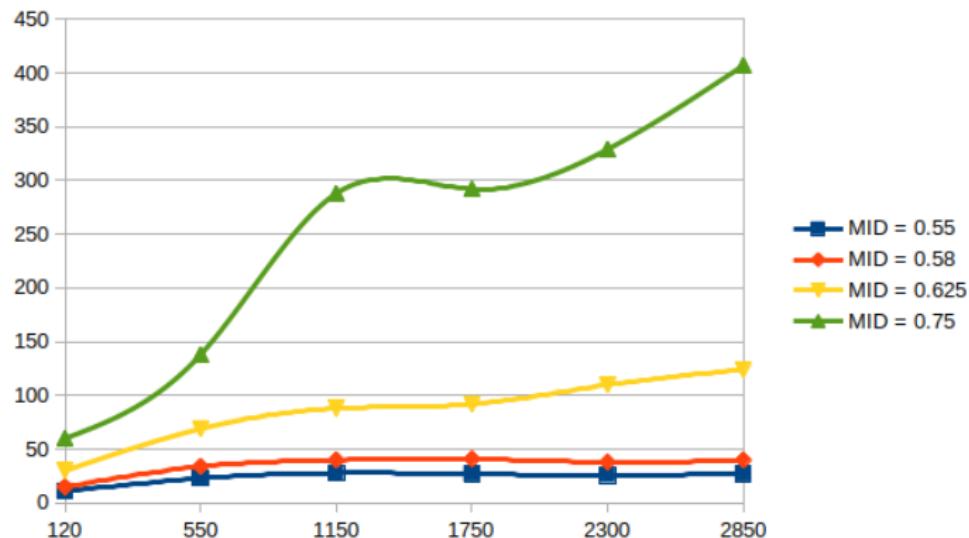


Результат



Кластеризация

MID - максимальная индивидуальная дистанция между элементами кластера. Эффективность повышается при увеличении числа объектов в исходной группе



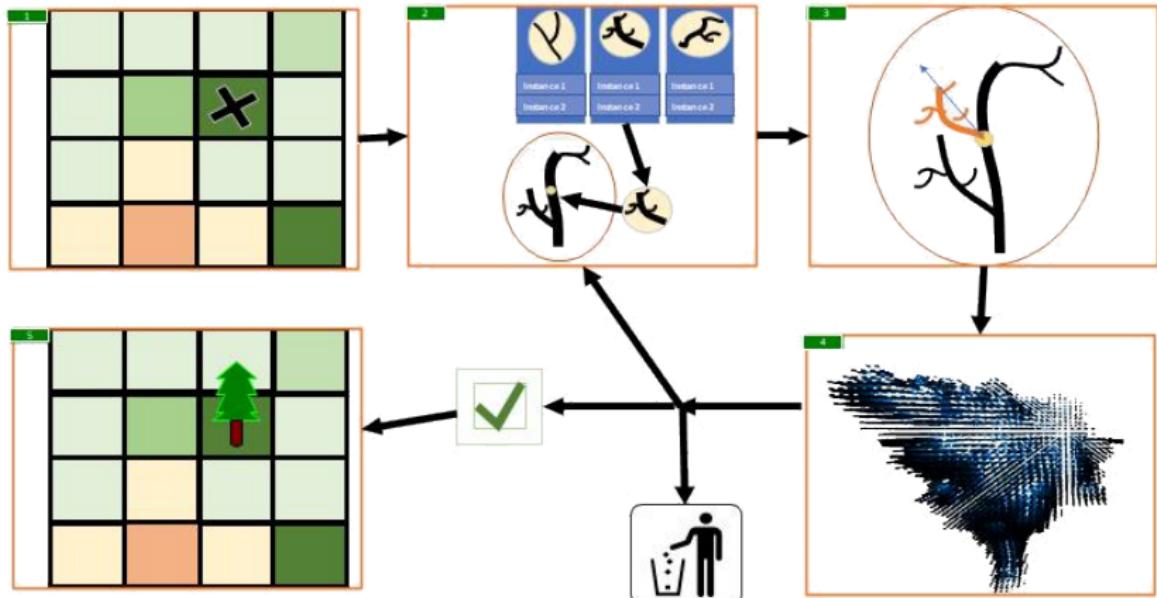
Кластеризация



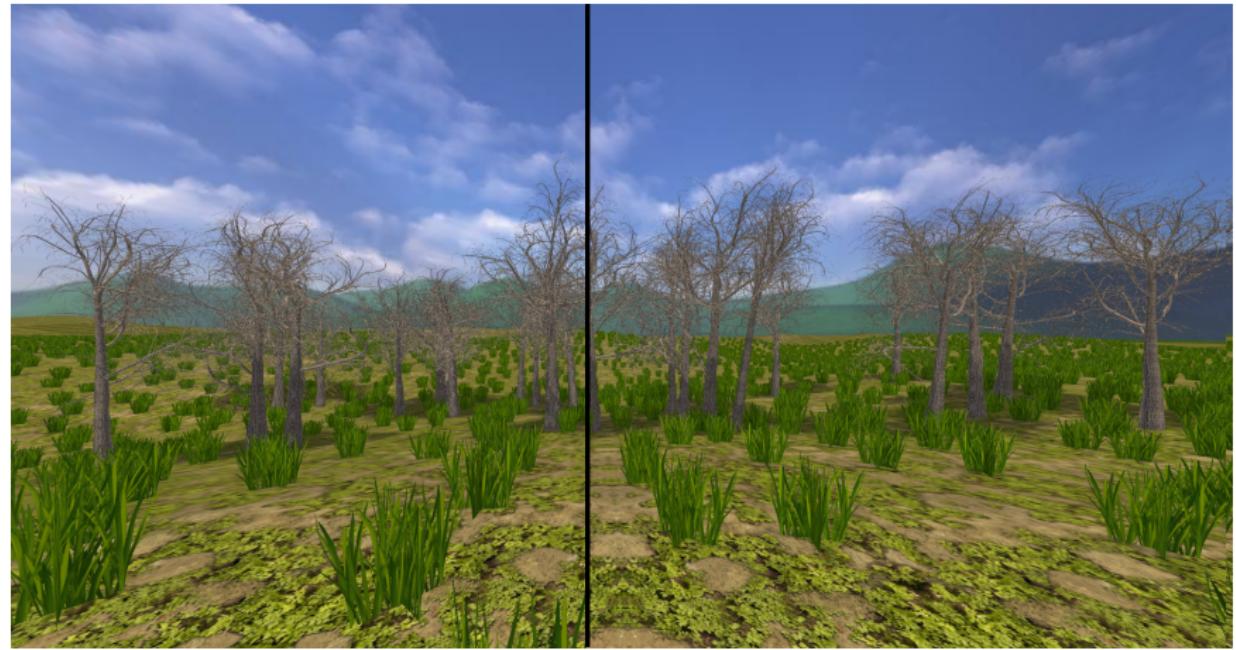
"Синтетические деревья"

- Генератор может работать очень медленно
- Деревья в больших группах могут быть менее реалистичны в деталях
- Можно создавать новые деревья из уже имеющихся моделей

"Синтетические деревья"



Результат



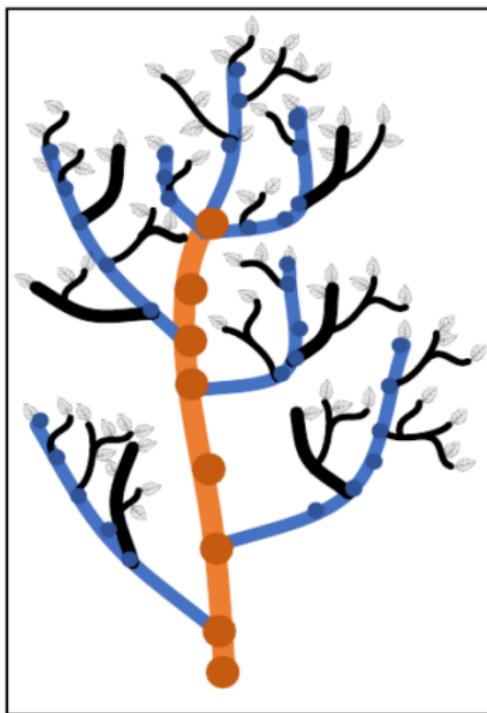
Преимущества

- 1) Сохраняется уникальность деревьев
- 2) Алгоритм не зависит от процедур генерации
- 3) Результатом работы алгоритма являются структуры данных, типичные для описания растительности на сцене. Для их рендера можно использовать уже существующие алгоритмы.

Конец

Спасибо за внимание!

Структура дерева



Кластеризация

Зная метрику расстояния между любыми двумя элементами множества, мы можем разделить его на кластеры одним из существующих общих алгоритмов

Алгоритм 1.6. Быстрая агломеративная кластеризация на основе редуктивности

- 1: инициализировать множество кластеров C_1 :
 $t := 1; C_t = \{\{x_1\}, \dots, \{x_\ell\}\};$
- 2: выбрать начальное значение параметра δ ;
- 3: $P(\delta) := \{(U, V) \mid U, V \in C_t, R(U, V) \leq \delta\};$
- 4: **для всех** $t = 2, \dots, \ell$ (t – номер итерации):
 - 5: **если** $P(\delta) = \emptyset$ **то**
 - 6: увеличить δ так, чтобы $P(\delta) \neq \emptyset$;
 - 7: найти в $P(\delta)$ пару ближайших кластеров:
 $(U, V) := \arg \min_{(U,V) \in P(\delta)} R(U, V);$
 $R_t := R(U, V);$
 - 8: изъять кластеры U и V , добавить слитый кластер $W = U \cup V$:
 $C_t := C_{t-1} \cup \{W\} \setminus \{U, V\};$
 - 9: **для всех** $S \in C_t$
 - 10: вычислить расстояние $R(W, S)$ по формуле Ланса-Уильямса;
 - 11: **если** $R(W, S) \leq \delta$ **то**
 - 12: $P(\delta) := P(\delta) \cup \{(W, S)\};$

Сбор статистики

```
struct TransformStat
{
    Normal *phi;
    Normal *psi;
    Normal *r;
    Normal *rot_angle;
};

You, 12 days ago | 1 author (You)
struct BranchStat
{
    DiscreteGeneral *typeStat = nullptr;
    TransformStat transformStat;
    bool valid = false;
};

You, 12 days ago | 1 author (You)
struct RootStat
{
    std::vector<DiscreteGeneral *> branchExistanceStat;
    BranchStat selfBranchStat;
    DiscreteGeneral *childBranchesClusterStat;
    int child_branches_count;
};

You, 18 days ago | 1 author (You)
struct FullStat
{
    DiscreteGeneral *rootClusterStat;
    std::vector<RootStat> rootStats;
    std::vector<BranchStat> branchStats;
    FullStat() {};
    ~FullStat();
};
```

Сжатие

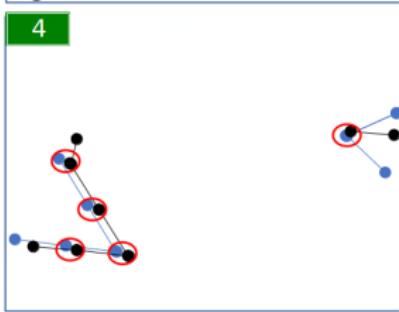
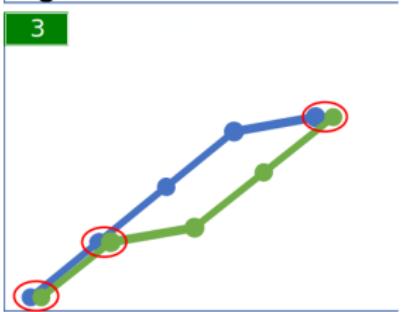
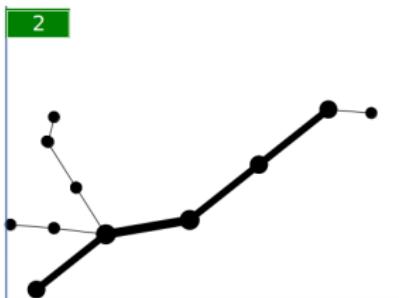
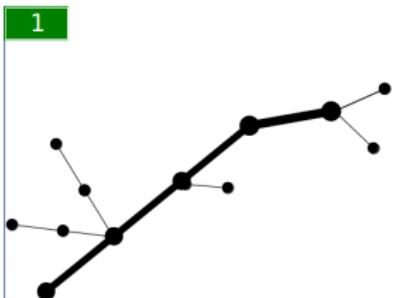
Table 1. Clustering time

Number of branches	Clustering time (seconds)
1250	11.86
2500	49.97
3750	111.09
5000	195.31
6250	291.78

Table 2. Compression ratio of the proposed method

Number of branches	$d_{max}=0.6$	$d_{max}=0.65$	$d_{max}=0.7$	$d_{max}=0.75$
1250	31.6	70.6	133.4	240.2
2500	30.2	83.5	192.6	357.7
3750	34.2	98.0	266.0	532.0
5000	34.6	103.2	275.1	550.2
6250	35.4	107.4	322.4	680.8

Node mapping



Метрика расстояния ("похожести")

Все ветки нормализуются - трансформируются так, чтобы их Bounding box был единичным кубом
Метрика расстояния:

$$M(B_1, B_2) = \min_{\alpha \in [0, 2\pi]} M_0(B_1, \text{rotate}(B_2, \vec{x}, \alpha))$$

$$M_0 = a * M_{struct} * (1 - b * R_{diff}) + (1 - a) * M_{light}$$

M_{struct} - метрика структурной "похожести"

M_{light} - метрика "похожести" затенения

R_{diff} - показатель разницы в толщине веток

Метрика структурной "похожести"

- Сопоставляем вершины веток друг с другом
- Начинаем с вершин собственно самих веток, сопоставлению подлежат вершины, находящиеся на расстоянии не более $\delta * \text{len}$

len - длина кратчайшей из сравниваемых веток

- Рекурсивно проводим процедуру сопоставления
Формально получаем множество пар:

$MJ = \{(j_1, j_2), j_1 \in B_1, j_2 \in B_2\}$ Тогда:

$$M_{struct} = \frac{1}{W_s} * \frac{\sum_{(j_1, j_2) \in MJ} (sw(j_1.level) + sw(j_2.level))}{\sum_{j_1 \in B_1} sw(j_1.level) + \sum_{j_2 \in B_2} sw(j_2.level)}$$

$sw(n)$ - константный массив весов для возможных уровней вершин, W_s - нормировочный множитель

Метрика структурной "похожести"

$$R_{diff} = \frac{1}{W_r} * \left(\sum_{(j_1, j_2) \in MJ} \int_0^{2\pi} \frac{|j_1.r(\alpha) - j_2.r(\alpha)|}{|j_1.r(\alpha) + j_2.r(\alpha)|} d\alpha \right)$$

W_r - нормировочный множитель

$$M_{light} = \iiint_V \frac{|V_1(x, y, z) - V_2(x, y, z)|}{V_1(x, y, z) + V_2(x, y, z)} dx dy dz$$