Сравнительный анализ методов синтеза текстур по образцу

А.Ю. Бабичев[1], В.А. Фролов[1,2]
andrey.babichev@graphics.cs.msu.ru|vfrolov@graphics.cs.msu.ru.
[1]Московский Государственный Университет имени М.В. Ломоносова, Москва, Россия;
[2]Институт прикладной математики имени М.В, Келдыша РАН.

В данной статье предлагается сравнительный анализ существующих методов синтеза текстур. Выявляются преимущества и недостатки существующих методов. В нашей работе мы руководствовались результатами обширного экспертного тестирования, в котором приняло участие более 25 человек на 20 принципиально разных текстурах. Это позволило нам сделать выводы о том, на каких типах текстур какие методы следует использовать. Наше исследование показало, что популярные в последнее время нейро-сетевые и статистические методы не являются лучшими ни по качеству синтеза, ни по скорости. Однако при этом они генерируют более разнообразные текстуры. С другой стороны, наиболее простой и быстрый метод переставления патчей показал лучшее качество и скорость. Таким образом, синтез текстур по образцу – одна из перспективных областей исследования, в которой различные подходы обладают разными преимуществами.

Ключевые слова: *компьютерная графика, синтез текстур, сравнение.*

Comparison of texture synthesis methods by sample

A.Y. Babichev[1], V.A. Frolov[1, 2].
andrey.babichev@graphics.cs.msu.ru|vfrolov@graphics.cs.msu.ru.
[1]Moscow State University, Moscow, Russia;
[2]Keldysh Institute of Applied Mathematics, Moscow, Russia;

This article provides a comparative analysis of existing texture synthesis methods. The advantages and disadvantages of the methods are revealed. In our work, we were guided by the results of extensive expert testing, which was attended by more than 25 people on 20 fundamentally different textures. This allowed us to draw conclusions about what types of textures which methods should be used. Our study showed that recently popular neural network and statistical methods are neither the best in terms of synthesis quality nor speed. However, they also generate more diverse textures. On the other hand, the easiest and fastest method for rearranging patches showed the best quality and speed. Thus, pattern synthesis of textures is one of the promising areas of research in which different approaches have different advantages.

Keywords: *computer graphics, texture synthesis, comparison.*

1. Введение

Текстуры – одна из основных компонент синтеза реалистичных изображений. С их помощью можно описать широкое многообразие поверхностей объектов, таких как минералы, шерсть, кожа, растения, а также различные ландшафты.

Текстуры можно получить множеством различных способов, например, с помощью сканирования фотографий или картин, нарисованных вручную. Однако оба этих подхода на практике являются чрезвычайно трудоёмкими, поскольку с фотографии необходимо удалять освещение, а границы текстур нужно обрабатывать специальным образом для удаления швов (появляющихся при циклическом наложении текстуры на объект).

Созданные с помощью синтеза по образцу изображения, могут быть любого размера, а при необходимости синтез можно настроить так, чтобы он не создавал швов (что может оказаться полезным при синтезе больших текстур из маленьких, так как можно создать новую маленькую текстуру и с помощью циклического наложения получить большую, которая будет обладать изначальным качеством). Область применения синтеза текстур по образцу весьма широка: заполнение дыр, генерация контента для машинного обучения, помощь художникам.

Принцип работы алгоритмов синтеза текстур можно описать следующим образом: на вход алгоритму подаётся исходная текстура, на ее основе создается новая (например, с помощью шума), которая затем с помощью какого-то метода изменяется так, чтобы визуально быть похожей на изначальную. Методы, с помощью которых создается новая текстура, и методы, с помощью которых полученная текстура становится похожей на изначальную, определяют все многообразие методов синтеза текстур по образцу:

В данной работе мы не только сравнили методы по качеству, но и постарались ответить на следующие вопросы:

1) Насколько хорошо метод сохраняет структурную организацию текстуры (характер/стиль узоров);
2) насколько качественной и оригинальной получается текстура;
3) может ли метод сохранять семантику (смысл) изображаемых объектов (например, если на исходной текстуре есть цветы, будут ли цветы или их части на сгенерированной текстуре);
4) создает ли метод швы при циклическом наложении.

2. Обзор методов синтеза текстур по образцу

В данной секции будут описаны методы синтеза, на которых проводилось тестирование.

2.1 Синтез текстур с помощью непараметрической выборки

Данный метод [2] синтезирует новую текстуру попиксельно, начиная с одного случайно выбранного пикселя. Для построения нового пикселя p, сначала определим $\omega(p)$, как квадратный патч со стороной длины ω с центром в p, а $d_{perc}(\omega_1, \omega_2)$ – как расстояние между патчами:

$$d_{perc} = d_{SSD} * G,$$

где d_{SSD} – среднеквадратическое расстояние, а G – гауссово ядро размерности два.

В первую очередь находится

$$\omega'_{best} = argmin_{\omega} d_{perc}(\omega(p), \omega),$$

где $\omega(p)$ – патч в синтезируемой текстуре с максимальной длиной стороны, а ω - патчи такой же длины, взятые из

изначального изображения. Также стоит отметить, что расстояние высчитывается только между уже построенными пикселями, то есть пиксели, значения которых еще не известны, не будут вносить вклад в расстояние.

Затем также находятся все остальные патчи ω', для которых выполнено $d_{perc}(\omega'_{best}, \omega') < \varepsilon$, где ε – некоторый небольшой порог (константа).

Наконец, значение искомого p может быть найдено, как среднее арифметическое всех центральных p, из найденных нами патчей ω'_{best} и ω'.

Данный процесс попиксельного построения текстуры будет продолжаться до тех пор, пока не будет построена текстура требуемого размера.

К проблемам данного алгоритма можно отнести то, что у него есть тенденция для некоторых текстур идти в «несмысловую» часть изображения и выращивать ложную текстуру, а также данный метод имеет свойство, что полученное изображение может оказаться изначальным со сдвигом.

С более подробным описанием алгоритма синтеза, а также его математическим обоснованием можно ознакомиться по [2].

2.2 Синтез текстур с помощью выборки на основе патчей

Данный алгоритм [1, 5] синтезирует изображение патчами слева направо и снизу вверх.

Сначала определим следующие переменные:

1) I_{in}, I_{out} – входное и синтезируемое изображение соответственно;
2) $B_{(x,y)}$ – патч, левый нижний угол которого имеет координаты (x, y) в I_{in};
3) B_k – *k-ый* патч, который будет вставлен в I_{out};
4) $E_{B_{(x,y)}}$ – граничная зона $B_{(x,y)}$, E_{out} – граничная зона I_{out};
5) ω_E – длина граничной зоны, некоторая задаваемая нами константа;
6) ω_B – длина квадратного патча, некоторая задаваемая нами константа.

Затем определим следующие выражения:

$$d(E_{B_k}, E^k_{out}) = [\tfrac{1}{A}\textstyle\sum_{i=1}^{A}(p^i_{B_k} - p^i_{out})^2]^{\frac{1}{2}}, –$$

расстояние между граничными зонами, где E^k_{out} – граничная зона для *k-ого* вставляемого патча, A – число пикселей в граничной зоне, $p^i_{B_k}$ и p^i_{out} – значения (одноканальные или трехканальные) *i-ого* пикселя в граничных зонах. Стоит отметить, что для вычисления расстояния будет учитываться только та граничная зона I_{out}, которая окажется на стыке при вставке B_k в I_{out}.

$$d_{max} = e[\tfrac{1}{A}\textstyle\sum_{i=1}^{A}(p^i_{out})^2]^{\frac{1}{2}}, –$$

максимально допустимое расстояние, где p^i_{out} – значения *i-ого* пикселя в граничной зоне E^k_{out}, e – некоторая константа не меньше нуля.

$$\Psi_B = \left\{B_{(x,y)} \middle| d\left(E_{B_{(x,y)}}, E^k_{out}\right) < d_{max}, B_{(x,y)} in\ I_{in}\right\}, –$$

множество патчей, граничная зона которых отличается менее чем на d_{max} от граничной зоны I_{out} после вставки в последнюю *k* патчей.

Алгоритм, наконец, будет выглядеть следующим образом:

1) выберем любой патч B_0 размера $\omega_B \times \omega_B$ из I_{in} и вставим его в левый нижний угол I_{out} и положим $k = 1$;
2) для данного *k* построим множество Ψ_B;
3) если Ψ_B оказалось пустым, положим $\Psi_B = \{B_{min}\}$, где B_{min} – патч, граничная зона которого является ближайшей к E^k_{out};
4) случайно выберем один патч из Ψ_B в качестве B_k;
5) вычислим поверхностную ошибку между B_k и патчами в I_{out}, с которыми он пересекается, и затем найдем путь минимальной стоимости вдоль области пересечения и сделаем его границей B_k;
6) вставим B_k в I_{out}, положим $k = k + 1$;
7) будем повторять шаги 2), 3), 4), 5), 6) до тех, пока I_{out} не окажется полностью заполненным.

Поверхностная ошибка вычисляется следующим образом:

$$err(E_{B_k}, E^k_{out})_{(i,j)} = (p^{(i,j)}_{B_k} - p^{(i,j)}_{out})^2,$$

а для поиска пути минимальной стоимости (если мы рассматриваем пересечение B_k и патча слева от него) для всей граничной зоны необходимо вычислить совокупную поверхностную ошибку для всей граничной зоны:

$$err_{cm}(i,j) = err(i,j) + min\big(err_{cm}(i-1, j-1), err_{cm}(i-1, j), err_{cm}(i-1, j+1)\big).$$

Искомым путем будет последовательность (i, j) до минимального err_{cm}, стоящего в последней строке.

Аналогичные действия проводятся для поиска пути минимальной стоимости при рассмотрении B_k и патча снизу от него.

С алгоритмом вычисления новой граничной зоны более подробно можно ознакомиться по [1], а с полным алгоритмом синтеза текстур патчами по [5].

Проблемы данного метода схожи с Texture Synthesis by Non-parametric Sampling: при синтезе определенных текстур полученная текстура может являться изначальной с некоторым сдвигом. Также вычисление новой граничной зоны может давать неприемлемый результат, если она имеет резкие цветовые переходы.

2.3 Синтез текстур с использованием сверточных нейросетей

Данный метод [4] основан на применении сверточных нейросетей, в качестве нейросети используется VGG-19, которая предобучена для распознавания объектов.

Для начала мы подадим наше исходное изображение в нейросеть и вычислим активацию для каждого слоя l нейросети. Каждая активация формирует некий набор отфильтрованных изображений, которые также известны как feature maps. Каждый слой с N_l фильтров будет иметь N_l feature maps, каждый из которых размера M_l (если сжать вектор в одно измерение). Таким образом, все feature maps могут храниться в матрице $F_l \in R^{N_l \times M_l}$, где F^l_{jk} – это активация *j-ого* фильтра в позиции k в слое l. Затем вычисляется матрица Грамма $G^l \in R^{N_l \times N_l}$, где $G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$, – матрица, которая характеризует свойства изображения.

Затем создадим новую текстуру, проинициализировав нашу изначальную текстуру некоторым шумом (например, белым). Далее суть метод состоит в сведению созданной текстуры к изначальной с помощью метода градиентного спуска.

Функцию ошибки можно будет вычислить, как

$$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} \omega_l E_l,$$

где ω_l – весовые коэффициенты вклада каждого слоя в ошибку, $\vec{x}$ и $\hat{\vec{x}}$ – оригинальное и генерируемое изображение соответственно, а E_l – вклад каждого *l-ого* слоя:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{ij} - \hat{G}^l_{ij})^2.$$

Производную E_l можно вычислить, как

$$\frac{\partial E_l}{\partial \hat{F}_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2}\left(\left(\hat{F}^l\right)^T (G^l - \hat{G}^l)_{ji}, & \hat{F}_{ij}^l > 0 \\ 0, & \hat{F}_{ij}^l < 0. \end{cases}$$

Наконец, градиент E_l и градиент $\mathcal{L}(\vec{x}, \vec{\hat{x}})$ могут быть вычислены с помощью метода обратного распространения ошибок, и весь последующий процесс сводится к минимизации ошибки. Подробнее о данном методе можно прочитать по [4].

У данного метода есть один существенный минус – неопределенность количества его итераций. Поскольку градиент, вычисляемый для минимизации, может быть очень мал по сравнению с текущей ошибкой, метод может очень медленно сходиться. Возможным решением данной проблемы может быть остановка минимизационного процесса при ошибке, равной некоторой величине, но это может повлечь за собой существенные различия между синтезируемой и изначальной текстурой.

2.4 Глубокая корреляция для синтеза текстур

Данный метод [6] во многом повторяет ранее рассмотренный синтез текстур с использованием сверточных нейросетей: он основан на использовании предобученной для распознавания объектов нейросети VGG-19, а также на оптимизации матрицы Грамма изначальной текстуры, проинициализированной некоторым шумом, с помощью метода обратного распространения ошибок.

Однако данный метод в функции ошибок помимо матрицы Грамма имеет еще три важных компонента. Пусть $F_{j,(n,m)}^l$ – это активация *j-ого* фильтра в позиции *(n,m)* в слое *l*. Тогда введем следующие переменные:

$$R_{j,(x,y)}^l = \sum_{n,m} \omega_{x,y} F_{j,(n,m)}^l F_{j,(n-x,m-y)}^l, -$$

набор матриц «глубокой взаимосвязи», где $x \in [-\frac{N_l}{2}; \frac{N_l}{2}]$, $y \in [-\frac{M_l}{2}; \frac{M_l}{2}]$, $\omega_{(x,y)} = [(N_l - x)(M_l - y)]^{-1}$.

$$E_{DCor}^l = \frac{1}{4} \sum_{x,y,j} (R_{j,(x,y)}^l - \hat{R}_{j,(x,y)}^l)^2, -$$

энергия «глубокой взаимосвязи» *l-ого* слоя.

$$E_{DCor} = \sum_l \omega_l^{Dc} E_{DCor,}^l -$$

полная энергия «глубокой взаимосвязи», где ω_l^{Dc} – весовые коэффициенты для данного слагаемого функции ошибок.

$$E_{Div}^l = \frac{1}{2} \sum_{n,m,j} (F_{j,(n,m)}^l - \hat{F}_{j,(n,m)}^l)^2, -$$

энергия «разнообразия» *l-ого* слоя.

$$E_{Div} = \sum_l \omega_l^D E_{D,}^l -$$

полная энергия «разнообразия», где ω_l^D – весовые коэффициенты для данного слагаемого функции ошибок.

$$E_{smooth}^l = \frac{1}{2\sigma} \sum_{n,m,j} log \sum_{\delta n, \delta m} \exp\left[-\sigma\left(F_{j,(n,m)}^l - F_{j,(n-\delta n, m-\delta m)}^l\right)^2\right], -$$

энергия «гладкости» *l-ого* слоя, где σ – некоторая константа, δn и δm – соседние индексы.

$$E_{smooth} = \sum_l \omega_l^S E_{smooth,}^l -$$

полная энергия «гладкости», где ω_l^S – весовые коэффициенты для данного слагаемого функции ошибок.

Таким образом, итоговая функция ошибок имеет следующий вид:

$$\mathcal{L} + E_{DCor} + E_{Div} + E_{smooth}.$$

Весь оставшийся алгоритм сводится к оптимизации функции ошибок методом обратного распространения ошибок. Более подробно о данном методе можно прочитать по [6].

Все проблемы этого метода схожи с проблемами синтеза текстур с использованием сверточных нейросетей.

2.5 Синтез текстур с помощью многоуровневой выборки

В основе этого метода [3, 7] лежит разложение изображения в пирамиды гауссиан. Сначала определим следующие переменные:

1) I_a, I_s – входное и выходное изображения;
2) G_a, G_s – пирамида гауссиан, построенная из I_a, I_s соответственно;
3) $N(p)$ – соседство вокруг пикселя p (например, все пиксели левее и выше p, расстояния не больше 2);
4) $G(L), G(L, x, y)$ – *L-ый* уровень пирамиды G и пиксель с координатами *(x,y)* в *G*;

Данный алгоритм можно описать следующими шагами:

1) проинициализируем I_s, как белый шум, имеющий размер, равный I_a;
2) разложим I_a, I_s в G_a, G_s;
3) положим $L = 1$ – уровень пирамиды, имеющий наименьшее разрешение;
4) пройдем по всем пикселям (x_s, y_s) пирамиды $G_s(L)$ (шаги 5) - 9));
5) положим, что N_s – соседство (x_s, y_s) уровня L пирамиды G_s, $N_a^{best} = null,\ C = null$;
6) пройдем по всем пикселям (x_a, y_a) пирамиды $G_a(L)$ (шаги 7)-8));
7) пусть N_a – соседство (x_a, y_a) уровня L пирамиды G_a;
8) если $d(N_a, N_s) > d(N_a^{best}, N_s)$, то $N_a^{best} = N_a$, $C = G_a(L, x_a, y_a)$, где d – некоторое расстояние, например среднеквадратичное;
9) $G_s(L, x_s, y_s) = C$, $L = L + 1$;
10) I_s – это результат сборки пирамиды G_s.

С более подробным описанием метода можно ознакомиться по [3, 7].

Поскольку данный метод основан на использовании только локальных соседств, он не сможет выдавать хороший результат на текстурах со сложными визуальными структурами (например, текстуры с прогрессивными изменениями в размере, цвете или ориентации или текстуры, детали которых зависят от их относительного положения), так как для них требуется анализ значительно большего соседства.

3. Выбор текстур

Для проверки способности методов сохранять структурированность, были отобраны текстуры, представляющие собой повторяющиеся несложные паттерны с малым количеством используемых цветов и без резких градиентных переходов. Такие текстуры выбирались с целью исключить влияние на синтез свойств текстуры, кроме её структурной организации. Так, на рис. 2, представлена текстура, состоящая из совершенно одинаковых деталей (кирпичей), а на рис. 1 - состоящая из сильно похожих деталей.

Чтобы узнать, насколько качественной и оригинальной получается текстура, синтезируемая тем или иным методом, были выбраны текстуры, состоящие из одноцветного фона с некоторой хаотичной картинкой поверх него. Такие

текстуры были выбраны с идеей, что любые методы синтеза должны отработать приемлемо на таких типах текстур в силу их простоты, но полученный результат может быть слишком похожим на изначальную текстуру или отличаться произвольными выбросами цветов, не свойственных данной текстуре. Результаты данной проверки можно увидеть на рис. 3 и рис. 4.

Для проверки возможности метода сохранить семантический смысл отбирались текстуры, представляющие собой хаотически повторенные, похожие друг на друга паттерны, в которых уже может использоваться произвольное количество цветов. Такие текстуры являются простыми узорами. С результатами данной проверки можно ознакомиться на рис. 5, рис. 6.

4. Результаты

Результаты синтеза всех рассмотренных методов можно увидеть в разделе иллюстрации, на рисунках 1,2,3,4,5,6.
Для выявления плюсов и минусов методов было проведено экспертное сравнение: предлагалось поставить оценку от 0 до 4 каждому методу на каждой текстуре из тестового набора. В таблице ниже приведены средние арифметические результатов с округлением в большую сторону.

	Структура	Оригиналь-ность /качество	Семантика	Швы
Patch-B	4	3	3	есть
No-Par	3	2	3	нет
Conv N	1	3	2	есть
DeepC	3	1	2	есть
MultiR	2	3	2	есть

5. Выводы

Как можно заметить по иллюстрациям, самым лучшим и стабильным методом оказался Patch-Based [1][5], на большинстве картинок он выдал приемлемый, если не лучший результат, однако к значительным минусам метода можно отнести то, что новая картинка получается полностью из исходной переставлением блоков, что значительно вредит оригинальности текстуры.

Обычный нейросетевой [4] метод плохо сохраняет структуру текстуры, но зато выдает довольно качественную и оригинальную картинку. Улучшение нейросетевого метода, Deep correlation [6], ведет себя наоборот: созданные им текстуры имеет схожую структуру с исходным изображением, однако они получаются плохого качества (с артефактами), а также сильно похожими на исходное. Также в силу вышеперечисленного следует и то, что оба метода плохо сохраняют семантику изображения.

Texture Synthesis by Non-parametric Sampling [2] хорошо сохраняет структуры текстур, однако в силу того, что этот метод синтезирует текстуру попиксельно, результаты его работы получаются слишком низкого качества, что является существенным минусом метода.

Текстуры, получаемые Multiresolution Sampling методом [3, 7], сохраняют неплохую структурированность, а также качество, но, несмотря на это, проигрывают другим методам в их сильных аспектах.

Таким образом, можно подвести итог, что для задач синтеза, не требующих абсолютно новой картинки лучше всего подходит Patch-Based метод [1, 5]; для задач заполнения небольших дыр в изображении наилучшим решением будет Non-parametric Sampling метод [2]. При требованиях создать новую оригинальную текстуру без требований структурированности следует использовать обычный нейросетевой метод [4] или Multiresolution Sampling метод [3, 7].

6. Иллюстрации

Рис. 1. Слева направо, сверху вниз: 1) исходное изображение, 2) Patch-Based, 3)Non-parametric Sampling, 4) Texture Synthesis Using Convolutional Neural Networks, 5) Deep Correlations for Texture Synthesis, 6) Multiresolution Sampling Procedure for Synthesis.

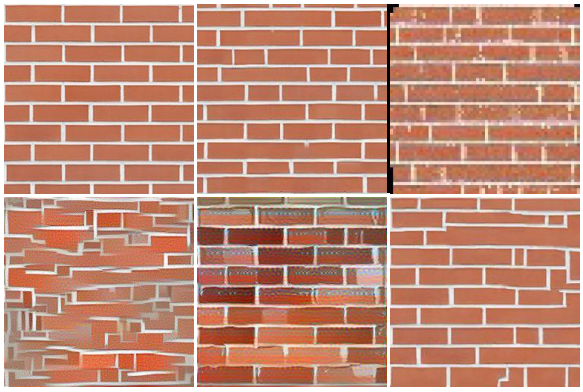

Рис. 2. Слева направо, сверху вниз: 1) исходное изображение, 2) Patch-Based, 3)Non-parametric Sampling, 4) Texture Synthesis Using Convolutional Neural Networks, 5) Deep Correlations for Texture Synthesis, 6) Multiresolution Sampling Procedure for Synthesis.

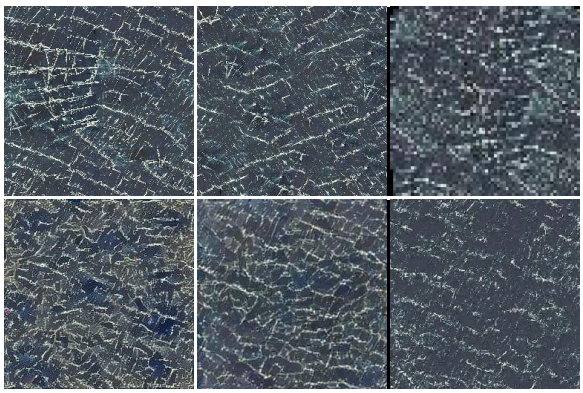

Рис. 3. Слева направо, сверху вниз: 1) исходное изображение, 2) Patch-Based, 3)Non-parametric Sampling, 4) Texture Synthesis Using Convolutional Neural Networks, 5) Deep Correlations for Texture Synthesis, 6) Multiresolution Sampling Procedure for Synthesis.

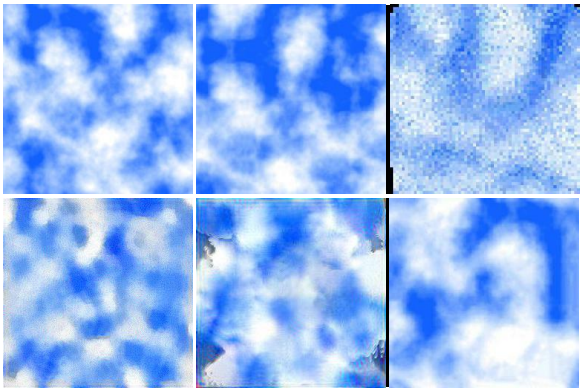

Рис. 4. Слева направо, сверху вниз: 1) исходное изображение, 2) Patch-Based, 3)Non-parametric Sampling, 4) Texture Synthesis Using Convolutional Neural Networks, 5) Deep Correlations for Texture Synthesis, 6) Multiresolution Sampling Procedure for Synthesis.

Рис. 5. Слева направо, сверху вниз: 1) исходное изображение, 2) Patch-Based, 3)Non-parametric Sampling, 4) Texture Synthesis Using Convolutional Neural Networks, 5) Deep Correlations for Texture Synthesis, 6) Multiresolution Sampling Procedure for Synthesis.

Рис. 6. Слева направо, сверху вниз: 1) исходное изображение, 2) Patch-Based, 3)Non-parametric Sampling, 4) Texture Synthesis Using Convolutional Neural Networks, 5) Deep Correlations for Texture Synthesis, 6) Multiresolution Sampling Procedure for Synthesis.

7. Благодарности

Работа выполнена при поддержке грантов РФФИ 18-31-20032 и 18-01-00569.

8. Литература

[1] Alexei A. Efros and William T. Freeman. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '01). ACM, New York, NY, USA, 341-346. DOI: https://doi.org/10.1145/383259.383296

[2] Alexei A. Efros and Thomas K. Leung. 1999. Texture Synthesis by Non-Parametric Sampling. In Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2 (ICCV '99), Vol. 2. IEEE Computer Society, Washington, DC, USA, 1033-.

[3] Jeremy S. De Bonet. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. In Proceedings of the 24th annual conference on Computer graphics and interactive techniques (SIGGRAPH '97). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 361-368. DOI: https://doi.org/10.1145/258734.258882

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. Texture synthesis using convolutional neural networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15), C. Cortes, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 1. MIT Press, Cambridge, MA, USA, 262-270.

[5] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. 2001. Real-time texture synthesis by patch-based sampling. ACM Trans. Graph. 20, 3 (July 2001), 127-150. DOI=http://dx.doi.org/10.1145/501786.501787

[6] Omry Sendik and Daniel Cohen-Or. 2017. Deep Correlations for Texture Synthesis. *ACM Trans. Graph.* 36, 5, pages. DOI: https://doi.org/10.1145/3015461

[7] Li-Yi Wei and Marc Levoy. 2000. Fast texture synthesis using tree-structured vector quantization. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 479-488. DOI=http://dx.doi.org/10.1145/344779.345009

С полными результатами можно ознакомиться по ссылке https://drive.google.com/file/d/1pHBgQk-XWa1U6LNRxU1P5oushcQDCZnH/view (методы даны в той же последовательности, что и на рисунках).

Об авторах

Бабичев Андрей Юрьевич, студент кафедры интеллектуальных информационных технологий факультета вычислительной математики и кибернетики Московского государственного университета. E-mail: andrey.babichev@graphics.cs.msu.ru.

Фролов Владимир Александрович, к.ф.-м.н., научный сотрудник кафедры интеллектуальных информационных технологий факультета вычислительной математики и кибернетики Московского государственного университета и Института Прикладной Математики имени М. В. Келдыша РАН. E-mail: vfrolov@graphics.cs.msu.ru.