

# Реализация и сравнение алгоритмов автоматической сборки пазлов

Курсовая работа  
студента 351 группы Синкевича А. А.

Саратовский национальный исследовательский  
государственный университет им. Н. Г. Чернышевского

Кафедра математической кибернетики  
и компьютерных наук

Научный руководитель: доцент, к. ф.-м. н. А. С. Иванова

2022

# Применение сборки пазлов

- ▶ Археология (сборка кусочков керамики);
- ▶ Реконструкция измельчённых документов и фотографий;
- ▶ Восстановление файлов изображений из фрагментов на диске;
- ▶ Перестановка, удаление объектов на изображении;
- ▶ Составление мозаики с плавными переходами;
- ▶ Восстановление речи, в записи которой переставлены части.

## Цели и задачи

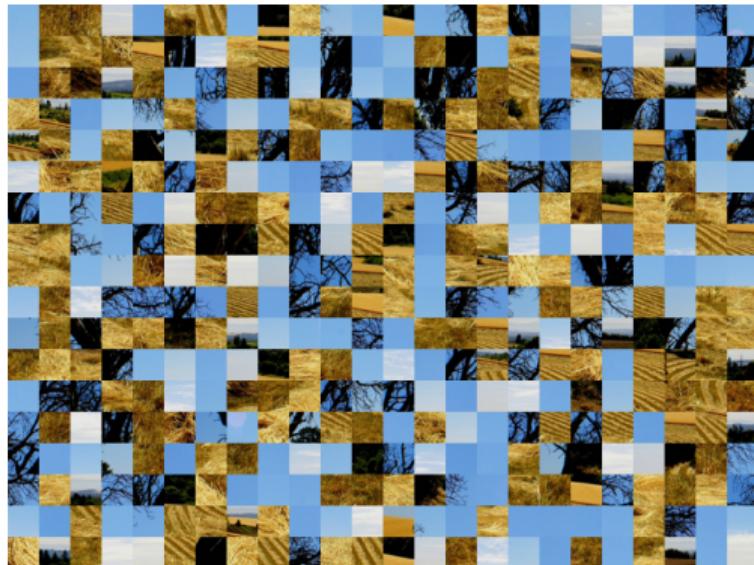
Целью данной работы является изучение различных алгоритмов автоматической сборки пазлов и сравнение их друг с другом по критериям времени работы и точности сборки. Для этого должны быть решены следующие задачи:

- ▶ реализация различных методов сравнения деталей;
- ▶ изучение и реализация выбранных алгоритмов сборки пазлов;
- ▶ создание приложения с графическим интерфейсом для выполнения и визуализации результатов работы алгоритмов;
- ▶ изучение методов оценки результатов;
- ▶ сравнение реализованных алгоритмов, использующих выбранные методы сравнения деталей, по различным критериям.

## Задача сборки пазлов

- ▶ Может использоваться форма фрагментов, цветовые данные или и то, и другое;
- ▶ Задача сборки прямоугольного изображения из квадратных деталей только с помощью цветовой информации:
  - ▶ Тип 1 — неизвестны позиции деталей, известна ориентация в пространстве (рассматривается в данной работе);
  - ▶ Тип 2 — неизвестны ни позиции, ни ориентации деталей;
  - ▶ Тип 3 — известны позиции деталей, неизвестна ориентация;
  - ▶ Типы 4, 5, 6, 7 — добавляется вторая сторона пазла.

# Пример



(a) Заданное изображение



(b) Результат сборки

## Решение

- ▶ Возможные решения — все перестановки ( $N!$ );
- ▶ Из-за неоднозначностей в определении совместимости деталей задача NP-трудная, возможно только приближённое решение;
- ▶ Необходимые части решения:
  - ▶ метод оценки совместимости смежных деталей,
  - ▶ стратегия сборки.

# Методы сравнения деталей

- ▶ SSD (Sum of Squared Differences, сумма квадратов разностей) в цветовом пространстве RGB или L\*a\*b\*;
- ▶ MGC (Mahalanobis Gradient Compatibility, совместимость градиентов Махalanобиса).

## Сумма квадратов разностей

- ▶ Этот метод показывает разность в цветах на границе деталей.
- ▶ Предположим, что деталь  $x_i$  находится слева от  $x_j$ , и они являются матрицами пикселей в заданном цветовом пространстве размера  $K \times K \times 3$ .
- ▶ Тогда SSD вычисляется как

$$C_{LR}(x_i, x_j) = \sqrt{\sum_{k=1}^K \sum_{c=1}^3 (x_i(k, K, c) - x_j(k, 1, c))^2}$$

- ▶ Чем меньше  $C$ , тем больше детали подходят друг другу.

# Совместимость градиентов Махalanобиса

- ▶ Этот метод показывает разность в градиентах цветов на границе деталей.
- ▶ Пусть  $x_i$  и  $x_j$  заданы так же, тогда градиенты в левой картинке:

$$G_L(k, c) = x_i(k, K, c) - x_i(k, K - 1, c)$$

- ▶ Далее вычисляется среднее значение градиента по каждому цветовому каналу:

$$\mu_L(c) = \frac{1}{K} \sum_{k=1}^K G_L(k, c)$$

## Совместимость градиентов Махalanобиса

- ▶ По  $\mu_L$  определяется матрица ковариаций цветовых каналов  $S_L$  размера  $3 \times 3$ ;
- ▶ Также вычисляется градиент из левой картинки в правую:

$$G_{LR}(k, c) = x_j(k, 1, c) - x_i(k, K, c)$$

- ▶ MGC слева направо:

$$D_{LR}(x_i, x_j) = \sqrt{\sum_{k=1}^K (G_{LR}(k) - \mu_L) S_L^{-1} (G_{LR}(k) - \mu_L)^T}$$

- ▶ Аналогично вычисляется MGC справа налево, и симметричный результат получается как  $C_{LR}(x_i, x_j) = D_{LR}(x_i, x_j) + D_{RL}(x_i, x_j)$ .

# Алгоритмы сборки

- ▶ Жадные;
- ▶ Алгоритм, основанный на многочастичном фильтре (метод Монте-Карло);
- ▶ Марковские случайные поля;
- ▶ Минимальные оставные деревья;
- ▶ Генетические алгоритмы;
- ▶ Алгоритм циклических ограничений.

# Генетический алгоритм

- ▶ Особь — одно из решений задачи.
- ▶ Популяция — набор особей, представляющих собой одно поколение.
- ▶ Поколение — один этап выполнения генетического алгоритма.
- ▶ Хромосома — информация, определяющая особь, т. е. матрица расстановки деталей в пазле;
- ▶ Функция приспособленности — оценка решения задачи данной хромосомой;
- ▶ Этапы алгоритма: создаётся начальная популяция, затем последовательно вычисляются новые поколения:
  - ▶ Оценка всех особей;
  - ▶ Отбор лучших особей в новое поколение («элитарность»);
  - ▶ Отбор пар особей из текущего поколения;
  - ▶ Скрещивание этих пар и составление нового поколения.

# Генетический алгоритм

- ▶ Функция неприспособленности — сумма несовместимостей всех смежных деталей. Пусть хромосома — матрица  $n \times m$ , тогда функция равна:

$$\sum_{i=1}^n \sum_{j=1}^{m-1} C_{LR}(x_{i,j}, x_{i,j+1}) + \sum_{i=1}^{n-1} \sum_{j=1}^m C_{UD}(x_{i,j}, x_{i+1,j})$$

- ▶ Начальная популяция — случайные перестановки;
- ▶ Отбор проводится методом рулетки: вероятность выбора особи  $p_i$  тем больше, чем меньше её значение функции неприспособленности  $f_i$ :

$$F_i = 0.95 + 0.9 \frac{\min_{j=1,\dots,P} f_j - f_i}{\max_{j=1,\dots,P} f_j - \min_{j=1,\dots,P} f_j}, p_i = \frac{F_i}{\sum_{j=1}^P F_j}$$

# Оператор скрещивания

1. Если есть грань, подходящая для фазы 1 (оба родителя соглашаются о детали), то эта деталь устанавливается и цикл повторяется;
2. Если есть грань, подходящая для фазы 2 (в одном из родителей есть деталь с «лучшим приятелем»), то эта деталь устанавливается и цикл повторяется;
3. Случайно выбирается грань, устанавливается наиболее совместимая деталь и цикл повторяется.

# Оператор скрещивания

- ▶ Цикл выполняется, пока есть свободные детали;
- ▶ Начальная деталь выбирается случайно, от неё «наращивается» полное изображение;
- ▶ Фаза 1: родители соглашаются о детали, если в них обоих в одном направлении от какой-то детали находится заданная;
- ▶ Две детали считаются «лучшими друзьями», если каждая из них считает другую наиболее совместимой с ней в определённом направлении;
- ▶ Фаза 2: деталь выбирается, если хотя бы в одном из предков она находится в нужном направлении от своего друга;
- ▶ При наличии в фазах 1 и 2 нескольких вариантов выбирается случайный, также в фазах 1 и 3 с небольшой вероятностью происходит мутация — выбирается случайная деталь.

# Оператор скрещивания — пример работы



(a) Родитель №1



(b) Родитель №2

# Оператор скрещивания — пример работы

# Оператор скрещивания — пример работы



Рис.: Результат работы

# Алгоритм циклических ограничений

- ▶ Определение лучших пар-кандидатов для каждой грани всех деталей;
- ▶ Вычисление маленьких циклов;
- ▶ Объединение матриц;
- ▶ Вырезание прямоугольного изображения, жадное заполнение пропусков.

## Вычисление маленьких циклов

- ▶ Составляются циклы 2-го порядка из 4 элементов так, чтобы каждая деталь считала смежные ей подходящими;
- ▶ Пока возможно, из циклов  $k$ -го порядка создаются циклы  $(k + 1)$ -го порядка:
  - ▶ Для всех пар определяются совместимости: если элементы (матрицы) размера  $i \times i$  пересекаются в области размера  $i \times (i - 1)$  или  $(i - 1) \times i$ , и непересекающиеся области не конфликтуют (нет общих деталей).
  - ▶ Из совместимых четвёрок составляются циклы  $(k + 1)$ -го порядка аналогично циклам 2-го порядка.

## Объединение матриц

- ▶ Используются маленькие циклы по убыванию размера;
- ▶ Последовательно объединяются все меньшие оставшиеся циклы без применения циклических ограничений;
- ▶ Если маленький цикл геометрически конфликтует с более крупной матрицей, цикл разбивается на составляющие подциклы более низкого порядка, а если два маленьких цикла одинаковой размерности конфликтуют, то используется цикл с меньшим средним значением несовместимости по всем смежным деталям в нём;
- ▶ Объединение не производится, если результирующая матрица больше пазла по какой-либо стороне или матрицы не связаны друг с другом (менее двух общих деталей);
- ▶ После всех слияний у матрицы-результата отрезаются почти пустые края и жадно заполняются пустые позиции.

# Методы оценки результатов

- ▶ Прямое сравнение: вычисляется доля деталей, находящихся в правильных позициях;
- ▶ Сравнение по соседям: средняя доля правильных соседей (с которыми деталь имеет общую грань) по всем деталям.

## Наборы изображений

- ▶ По 20 изображений с  $N = 432$ ,  $N = 540$  и  $N = 805$  деталями, и по 3 изображения с  $N = 2360$  и  $N = 3300$ , при размере деталей  $K = 28$ .
- ▶ Для некоторых изображений камера идеально выровнена по линии горизонта, а края изображения (например, границы зданий) точно совпадают с краями пазла. Некоторые детали содержат недостаточно информации (однородные области, такие как небо, вода и снег), а другие содержат повторяющиеся текстуры (искусственные поверхности и окна).
- ▶ Метрики попарной совместимости деталей возвращают много ложноположительных и ложноотрицательных результатов для этих изображений.

# Параметры генетического алгоритма

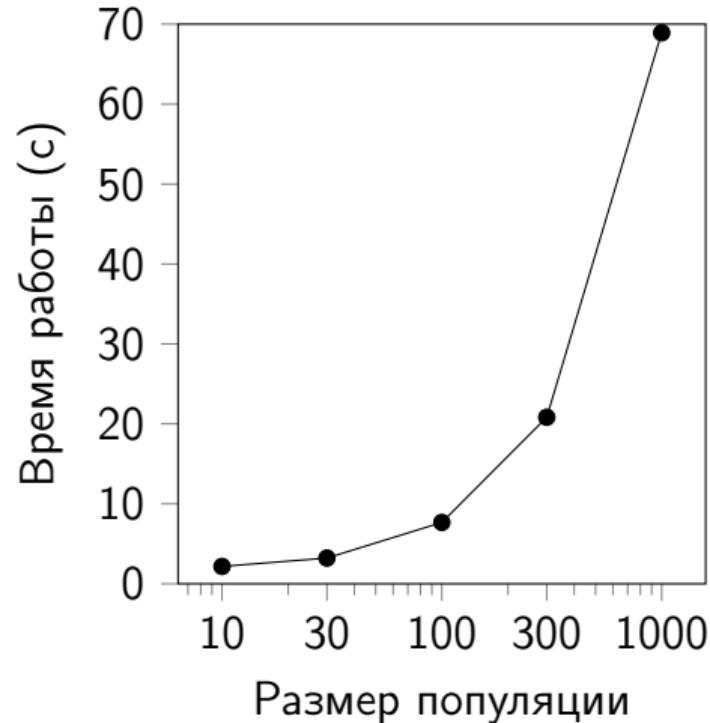
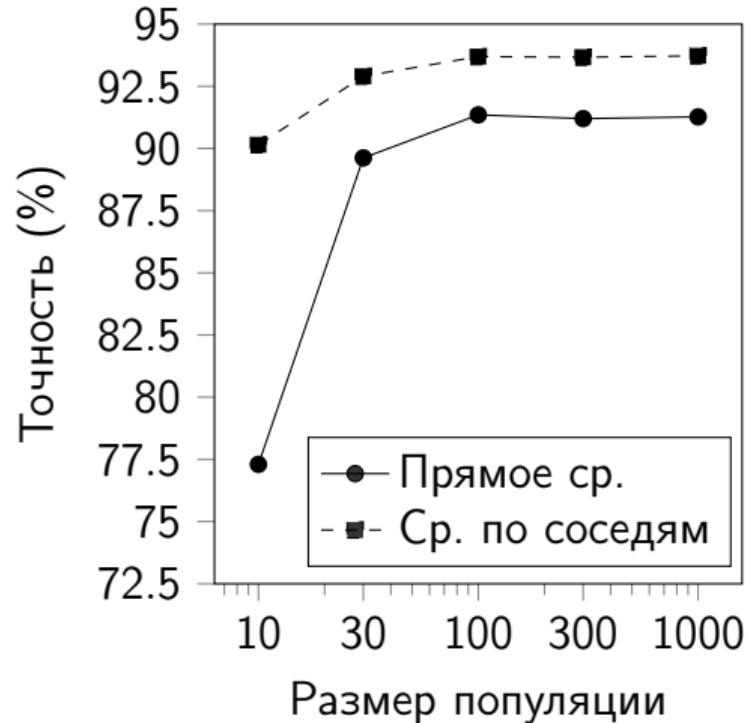


Рис.: Точность и время работы в зависимости от размера популяции на наборе изображений с  $N = 3300$ , используется LAB SSD

## Параметры генетического алгоритма

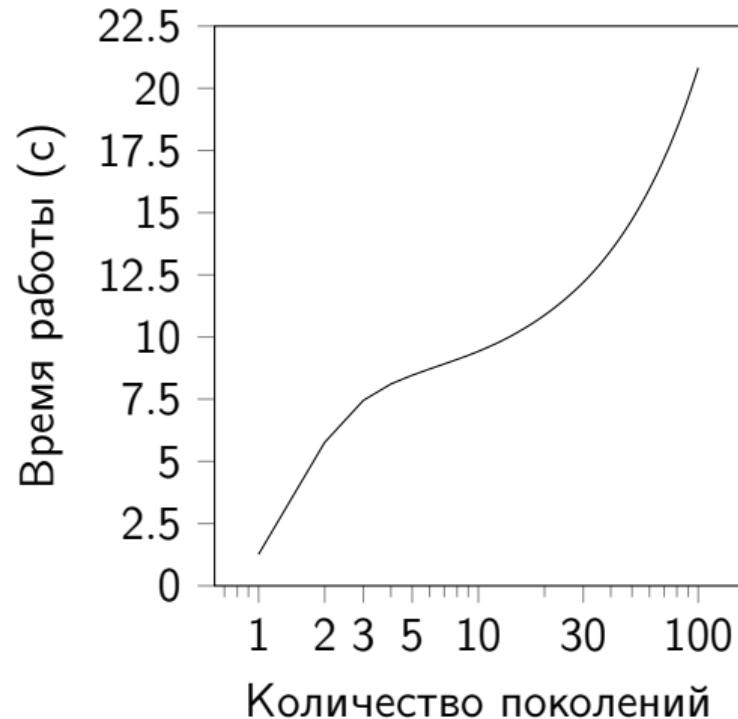
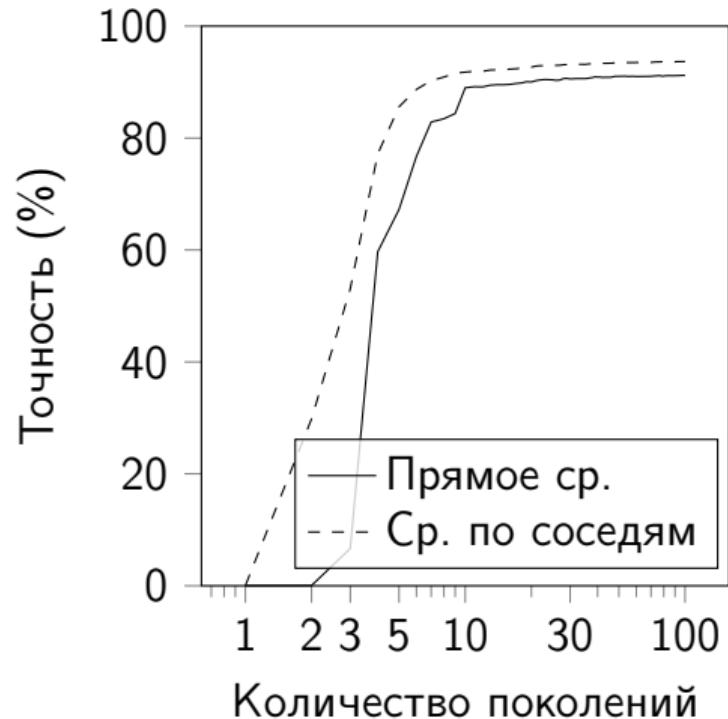


Рис.: Точность и время работы в зависимости от количества поколений на наборе изображений с  $N = 3300$ , используется LAB SSD

# Параметры генетического алгоритма

- ▶ Метод сравнения деталей: MGC;
- ▶ Количество поколений:  $G = 30$ ;
- ▶ Размер популяции:  $P = 100$ .

# Сравнение алгоритмов

Таблица: Точность (по методу сравнения по соседям, %) всех алгоритмов

$N$	Генетический алгоритм (работа Шоломон и др.)	Алгоритм циклических ограничений (работа Сон и др.)	Генетический алгоритм (данная работа)	Алгоритм циклических ограничений (данная работа)
432	96.16	95.5	95.72	94.69
540	95.96	95.2	96.29	92.67
805	96.26	94.9	95.90	92.66
2360	88.86	96.4	94.76	90.15
3300	92.76	96.4	96.62	94.06

# Сравнение алгоритмов

Таблица: Время работы (в секундах) всех алгоритмов

$N$	Генетический алгоритм (работа Шоломон и др.)	Алгоритм циклических ограничений (работа Сон и др.)	Генетический алгоритм (данная работа)	Алгоритм циклических ограничений (данная работа)
432	48.73	140	0.3228	0.3543
540	64.06	Н/Д	0.4312	0.4745
805	116.18	Н/Д	0.8580	0.9435
2360	1056	Н/Д	7.5897	11.6076
3300	1814.4	Н/Д	14.2809	23.6131

# Реализация

- ▶ Проект состоит из двух частей: библиотеки, содержащей описанные алгоритмы и вспомогательные функции, и приложения, реализующего графический интерфейс для выполнения этих алгоритмов.
- ▶ Для написания программы был выбран язык Rust из-за сочетания высокой производительности и безопасной работы с памятью и потоками.
- ▶ Также для обработки статистики выполнения алгоритмов, создания таблиц и графиков использовался язык Python в интерактивной среде Jupyter Notebook.

# Интерфейс

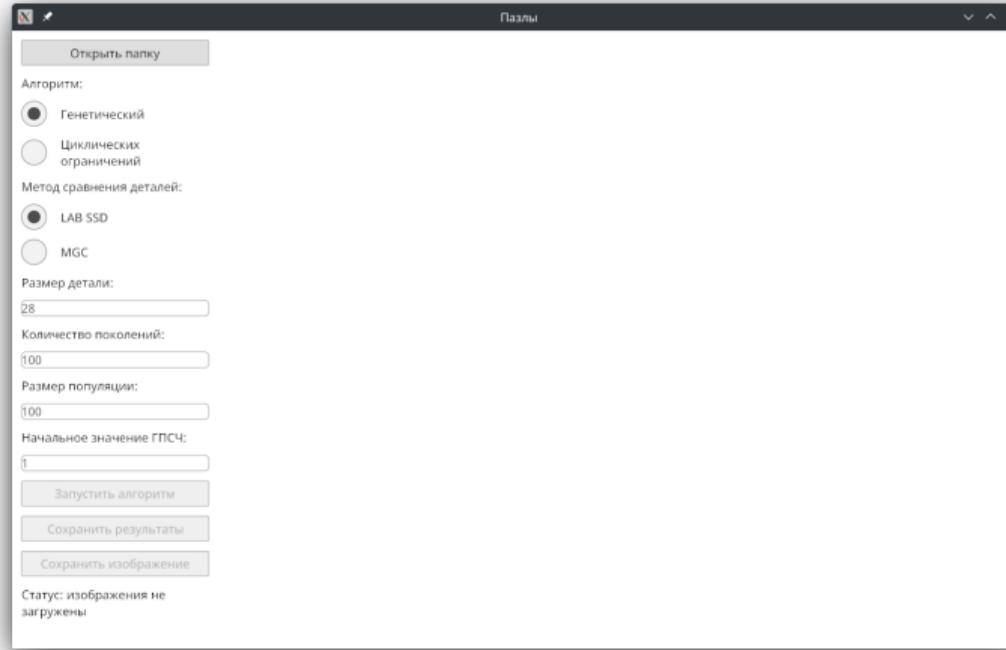


Рис.: Окно приложения после запуска

# Интерфейс

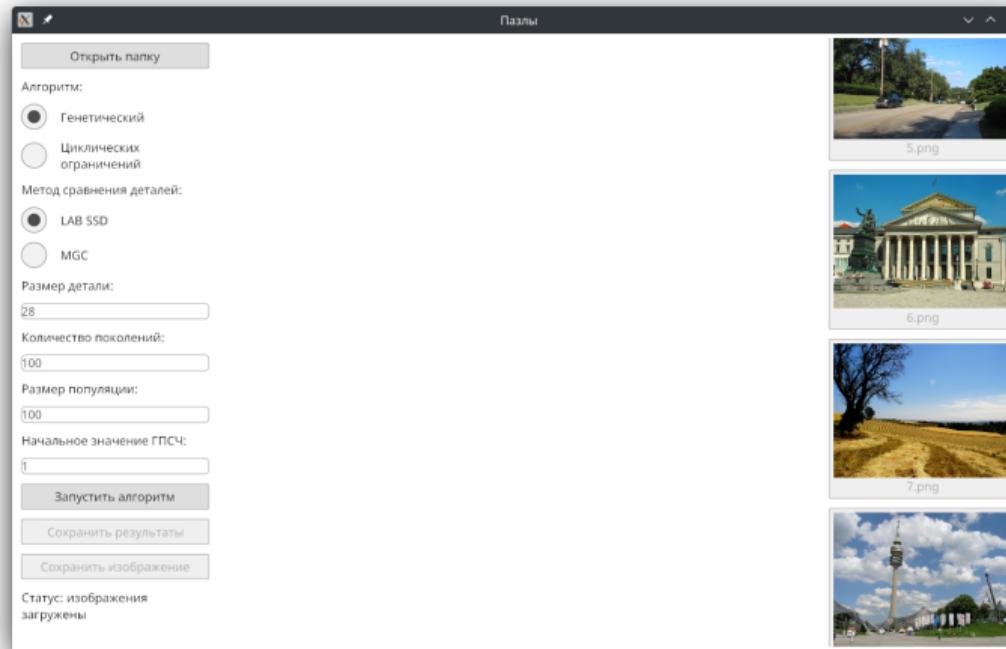


Рис.: Окно приложения после загрузки изображений

# Интерфейс

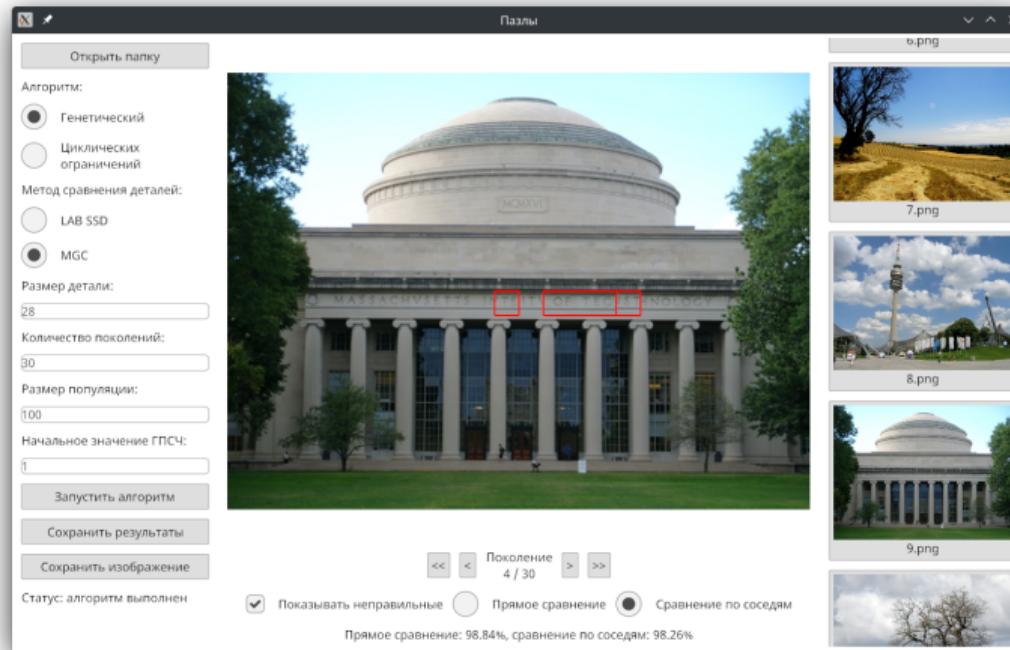


Рис.: Окно приложения после выполнения алгоритма — просмотр результатов