

Детектирование неба на фотографиях

Выполнил: Попов К.В.

Научный руководитель: Вахитов А.Т.

Цели

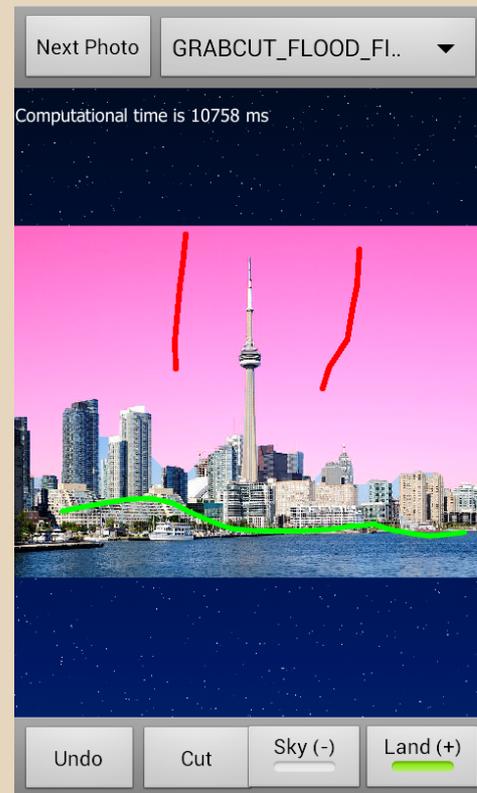
- Анализ существующих алгоритмов сегментации изображений, качества их работы и производительности
- Разработка алгоритма автоматического определения неба на фотографиях
- Анализ производительности
- Разработка мобильного приложения

Алгоритмы сегментации

- K-means
- Watershed
- Grabcut

Алгоритм

1. Фильтр высоких частот (High-pass)
2. Отсечение по порогу (интенсивность)
3. Вычисление матрицы расстояний
4. Отсечение по порогу (расстояние)
5. Поиск максимумов
6. Вычисление связанных регионов
7. Получение матрицы пикселей, наиболее вероятно принадлежащих небу
8. Grabcut



Этапы работы алгоритма

Next Photo Preview STEP_BY_.. ▾

Computational time is 259 ms



Undo Cut Sky (-) Land (+)

This panel shows the initial step of the algorithm. It features a control bar at the top with 'Next Photo', 'Preview', and 'STEP_BY_..' (with a dropdown arrow). Below this, a blue bar indicates 'Computational time is 259 ms'. The main image is a grayscale photograph of a mountain range reflected in a lake. At the bottom, a light blue bar contains a control panel with 'Undo', 'Cut', 'Sky (-)' (highlighted with a green underline), and 'Land (+)' (with a white underline).

Next Photo Preview STEP_BY_.. ▾

Computational time is 301 ms

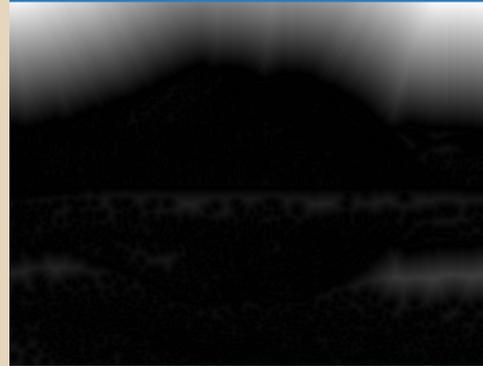


Undo Cut Sky (-) Land (+)

This panel shows the second step of the algorithm. The control bar and blue bar are identical to the first panel. The main image is a grayscale edge detection result of the original photo, showing the outlines of the mountain and lake. The bottom control panel is identical to the first panel.

Next Photo Preview STEP_BY_.. ▾

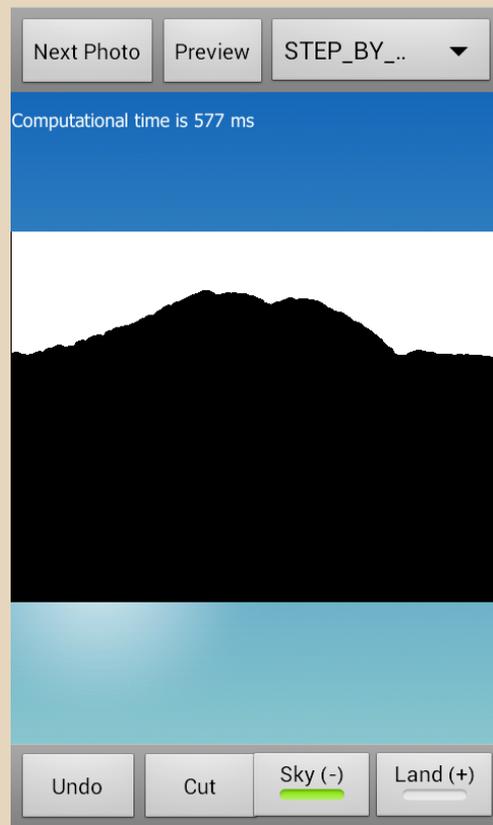
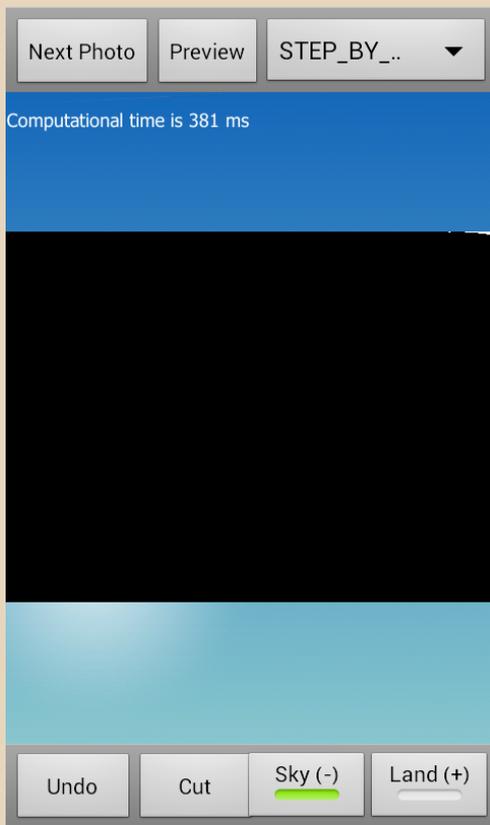
Computational time is 262 ms



Undo Cut Sky (-) Land (+)

This panel shows the third step of the algorithm. The control bar and blue bar are identical to the first panel. The main image is a blurred version of the edge detection result from the second panel. The bottom control panel is identical to the first panel.

Этапы работы алгоритма



Результат

Next Photo Preview STEP_BY_.. ▾

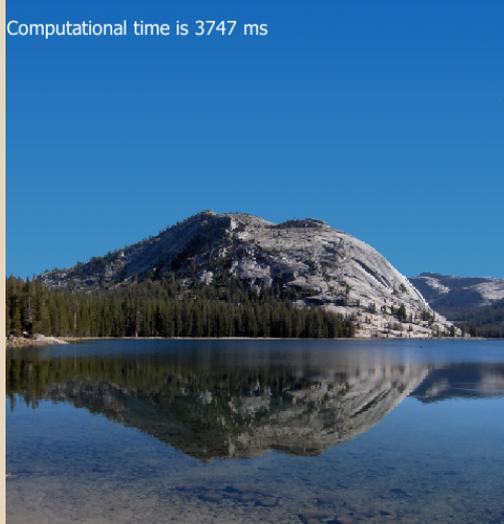
Computational time is 3747 ms



Undo Cut Sky (-) Land (+)

Next Photo Preview STEP_BY_.. ▾

Computational time is 3747 ms



Undo Cut Sky (-) Land (+)

Next Photo Preview STEP_BY_.. ▾

Computational time is 3747 ms



Undo Cut Sky (-) Land (+)

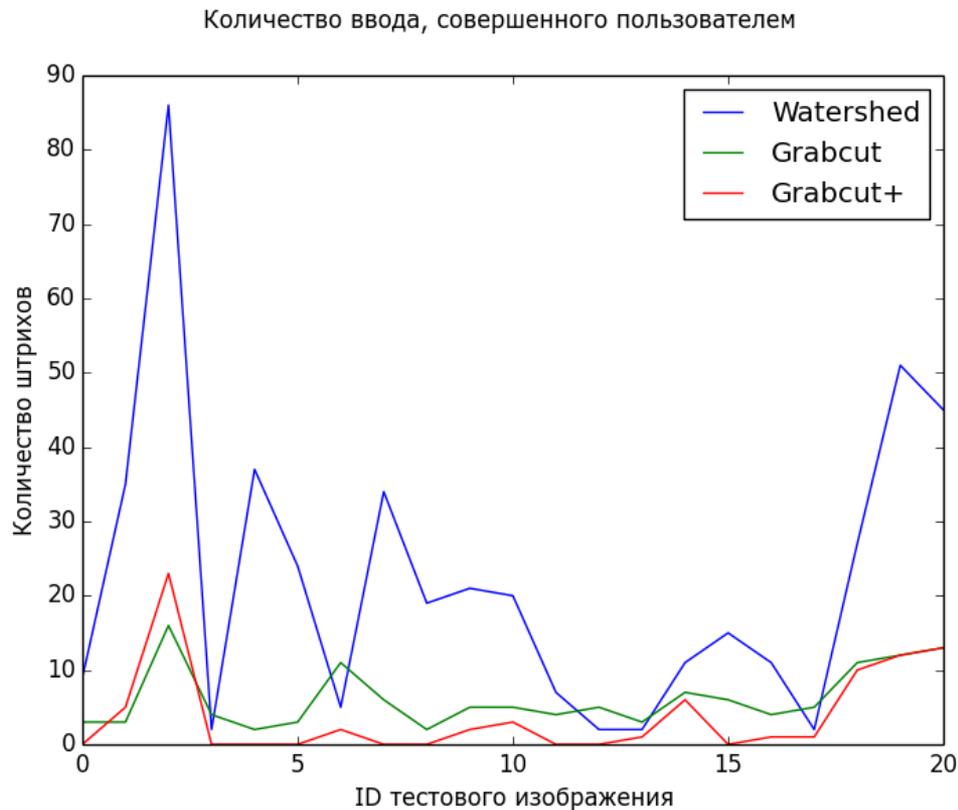
Сравнение результатов

Тестирование на испытуемых.

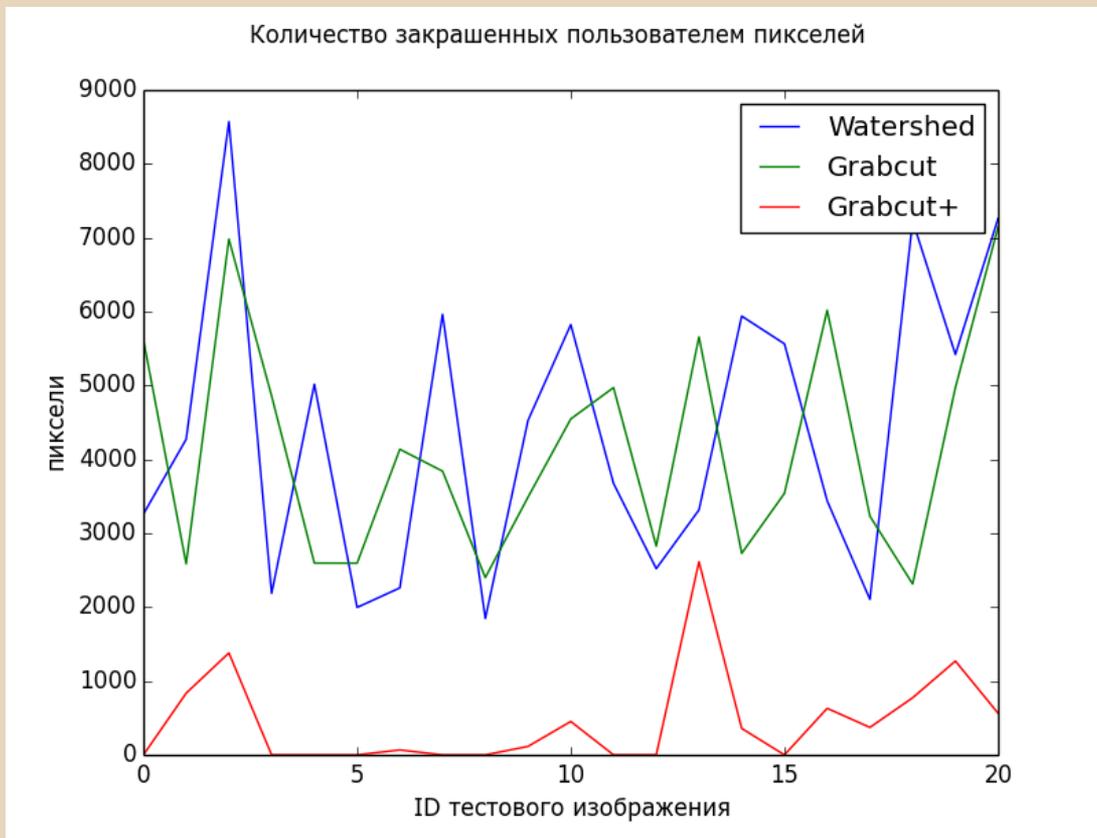
Собираемые параметры:

- Количество штрихов
- Количество закрашенных пикселей
- Время работы

Сравнение результатов

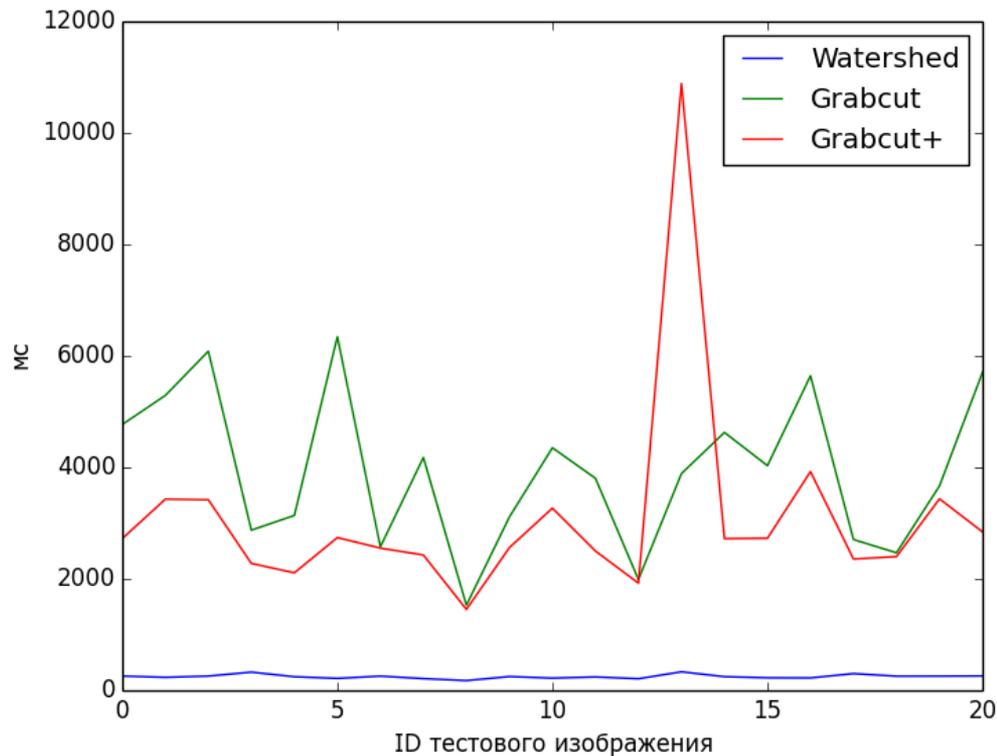


Сравнение результатов



Сравнение результатов

Среднее время обработки ввода



Математическая модель неба

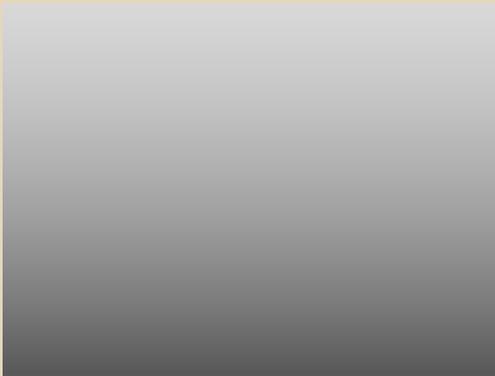
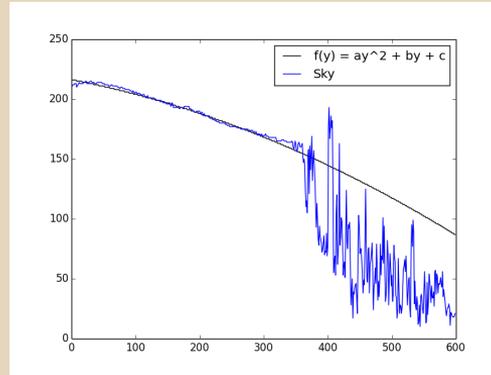
Использовался МНК

В качестве данных - ввод пользователя

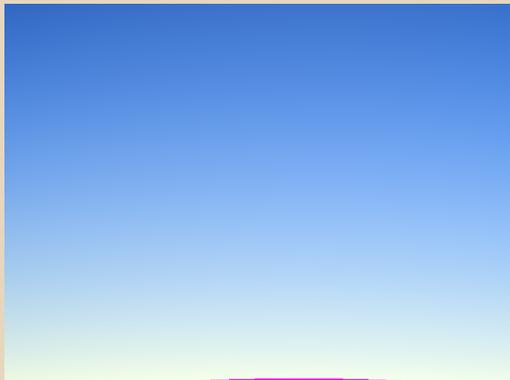
Рассматривалось 3 вида моделей:

- $f(y) = ay^2 + by + c$ (GRAY)
- $f(y) = ay^2 + by + c$ (RGB)
- $f(x,y) = ax^2 + bxy + cy^2 + dx + ky + l$ (RGB)

$$f(y) = ay^2 + by + c \text{ (GRAY)}$$



$$f(x,y) = ax^2 + bxy + cy^2 + dx + ky + l \text{ (RGB)}$$



Java vs Native код

- Bitmap - хранится в native памяти
- Каждый вызов `getPixel()` - вызов JNI методов
- Можно за один вызов JNI с помощью `getPixels()` получить сразу массив пикселей
- Ускорение в 30 раз на моем проекте

Результаты

1. Получен алгоритм, умеющий автоматически распознавать и удалять небо
2. Разработано приложение для OS Android
3. Выработаны критерии сравнения подобных алгоритмов
4. Опробован подход с моделированием неба