

# Детектирование неба на фотографиях

Выполнил: Попов К.В.

Научный руководитель: Вахитов А.Т.

# Цели

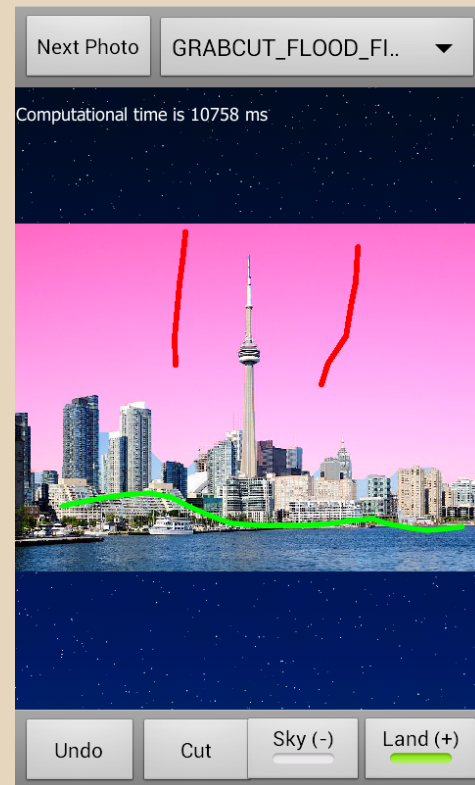
- Анализ существующих алгоритмов сегментации изображений, качества их работы и производительности
- Разработка алгоритма автоматического определения неба на фотографиях
- Анализ производительности
- Разработка мобильного приложения

# 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 input image. The interface includes a top bar with 'Next Photo', 'Preview', and 'STEP\_BY\_..' (dropdown). Below is a blue header with the text 'Computational time is 259 ms'. The main area displays a grayscale photograph of a mountain range reflected in a lake. At the bottom, there is a light blue bar and a control bar with 'Undo', 'Cut', 'Sky (-)' (highlighted with a green underline), and 'Land (+)' (underlined).

Next Photo Preview STEP\_BY\_.. ▾

Computational time is 301 ms

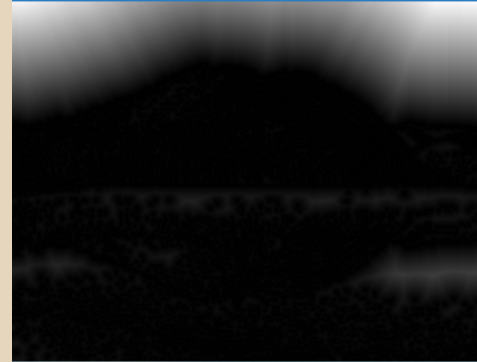


Undo Cut Sky (-) Land (+)

This panel shows the edge detection result of the image. The interface is identical to the first panel, but the main image area now displays a binary edge detection result where the mountain and water features are represented by white lines on a black background. The text 'Computational time is 301 ms' is shown in the blue header. The control bar at the bottom remains the same.

Next Photo Preview STEP\_BY\_.. ▾

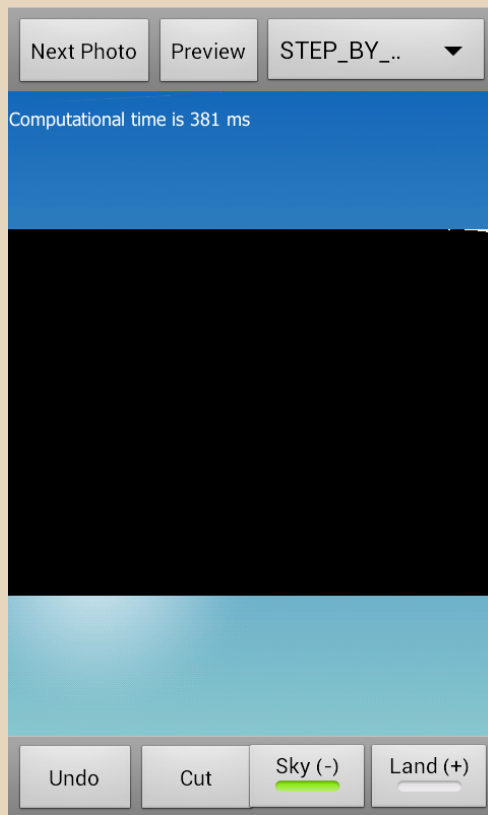
Computational time is 262 ms



Undo Cut Sky (-) Land (+)

This panel shows a blurred version of the edge detection result. The interface is identical to the previous panels, but the main image area now displays a blurred version of the edge detection result, showing a soft gradient. The text 'Computational time is 262 ms' is shown in the blue header. The control bar at the bottom remains the same.


# Этапы работы алгоритма



# Результат

Next Photo Preview STEP\_BY\_.. ▾

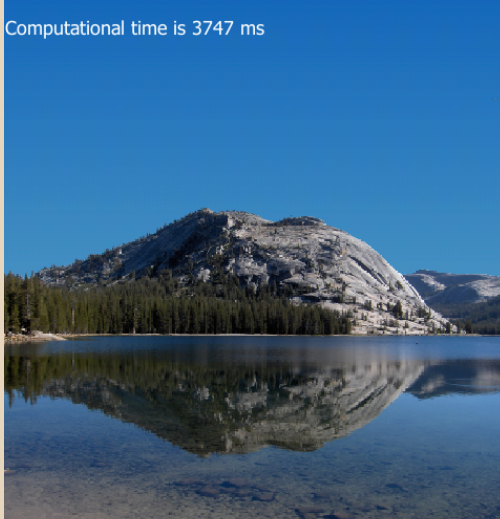
Computational time is 3747 ms



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

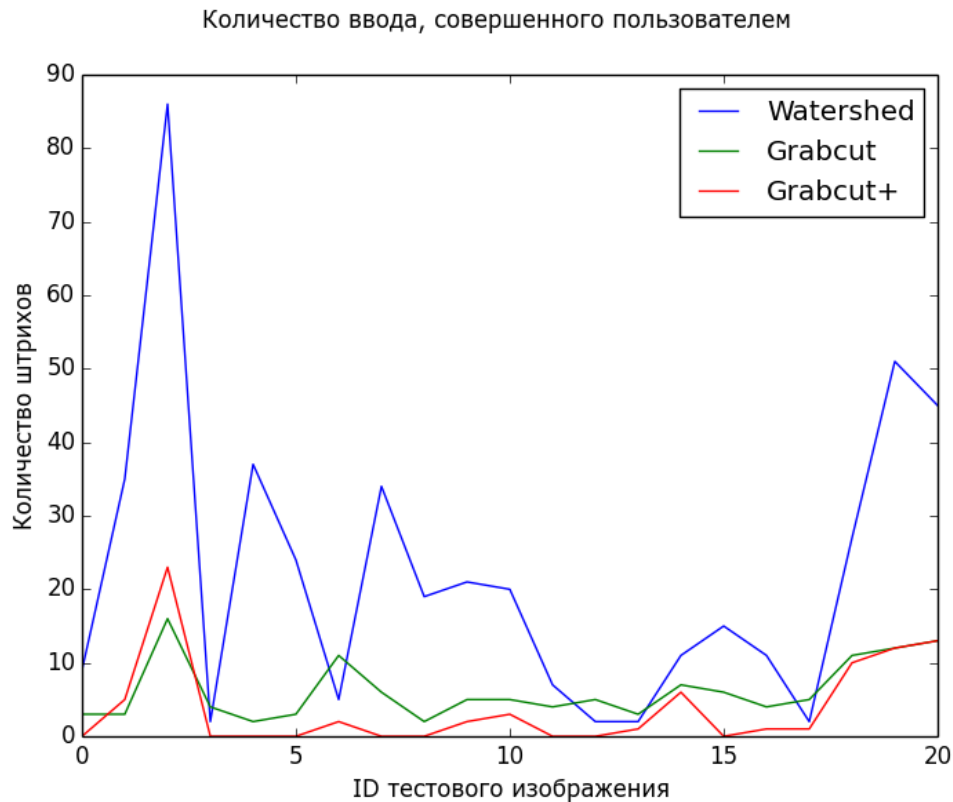
Тестирование на испытуемых.

Собираемые параметры:

- Количество штрихов
- Количество закрашенных пикселей
- Время работы



# Сравнение результатов

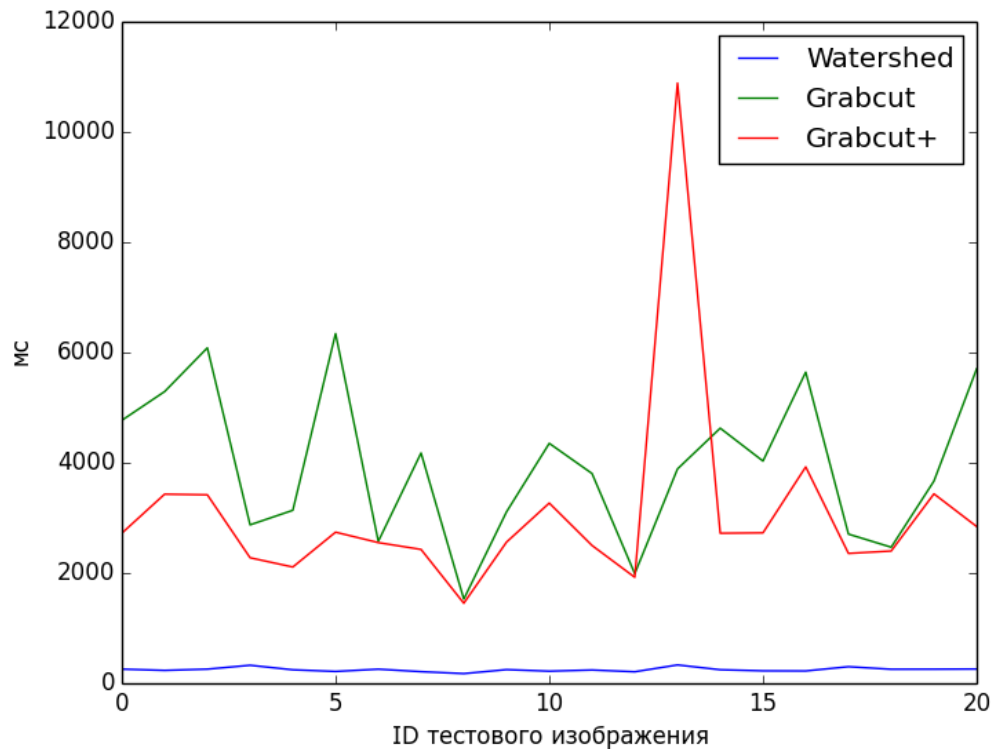


# Сравнение результатов



# Сравнение результатов

Среднее время обработки ввода



# Математическая модель неба

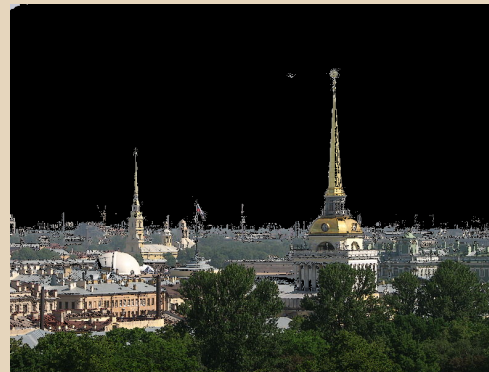
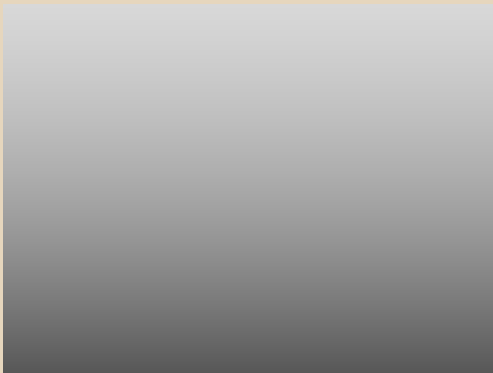
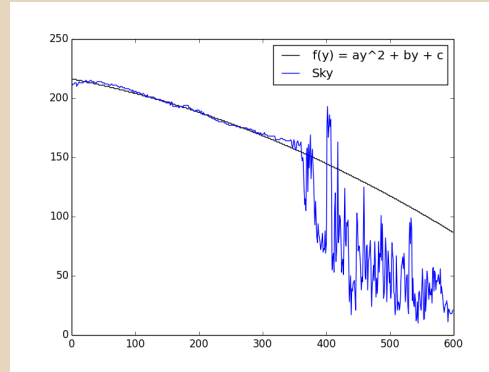
Использовался МНК

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