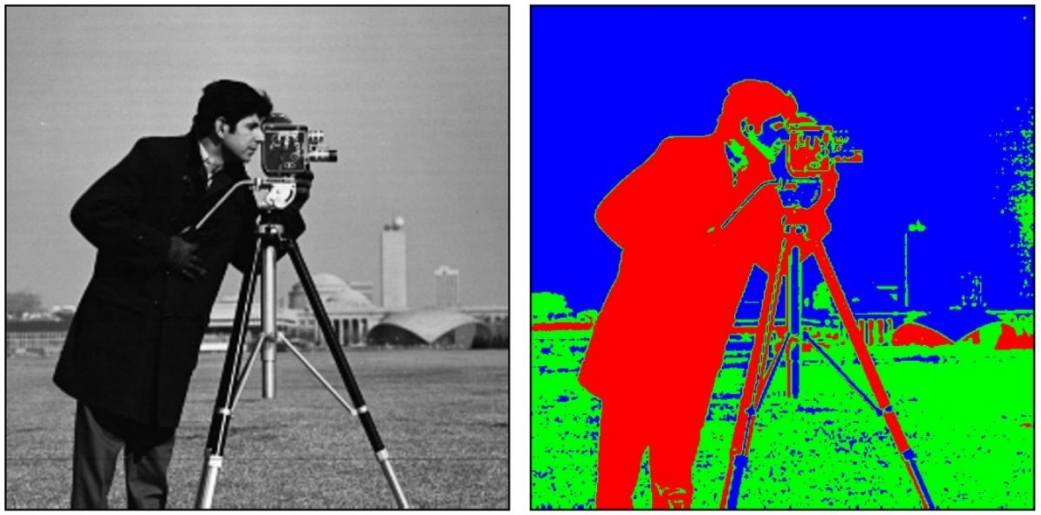
Исследование методов сегментации изображений

Артем Скребков, Владислав Виноградов,
Дмитрий Кручинин, Евгений Долотов

Руководитель: Козинов Е.А.

Сегментация изображения

- ***Сегментация*** – процесс разделения изображения на области, каждая из которых содержит пиксели, обладающие общими визуальными характеристиками.



Математическая постановка задачи

$$\varphi: I \to M$$

$I = \left(I(x,y)\right)_{\substack{0 \le x < w \\ 0 \le y < h}}$ – исходное изображение,

$M = \left(M(x,y)\right)_{\substack{0 \le x < w \\ 0 \le y < h}}$ – матрица меток,

где $M(x,y) \epsilon \{0, \dots, k-1\}$

МЕТОД К-СРЕДНИХ

Постановка задачи

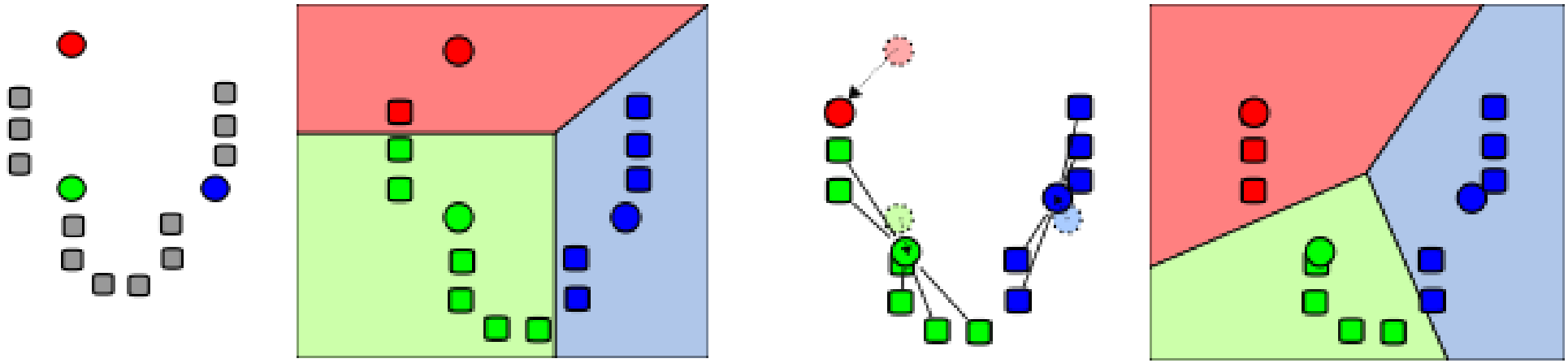
- ***Входные данные:***
  - I –изображение в формате RGB(каждый пиксель описывается трехмерным вектором интенсивностей), w, h – ширина и высота изображения
  - k – количество сегментов, которые необходимо получить
- ***Выходные данные:***
  - M – матрица меток размера $h \times w$, метка соответствует номеру сегмента

Схема алгоритма k-средних

1. Выбрать центры кластеров (случайно)
2. Определить кластер, которому принадлежит каждый пиксель изображения
3. Пересчитать центры кластеров посредством усреднения вектора интенсивностей в кластере
4. Повторить 2-3, пока центры кластеров не перестанут смещаться

Пример работы алгоритма для двумерных векторов

1 2 3 2



Модификация k-средних++

- Отличается от k-средних только начальным выбором центров кластеров:
  - Первый центр выбирать случайно
  - Каждый следующий центр выбирать с вероятностью $\frac{D^2(x)}{\sum_{\overline{x}\in X} D^2(\overline{x})}$, где $D(x)$ – расстояние от точки x до ближайшего из уже выбранных центров, X – множество пикселей
- Чем дальше точка от выбранных центров, тем больше вероятность ее выбора в качестве нового центра

Тестовая инфраструктура

- ОС: Windows 7 x64
- Процессор: Intel® Core™ i5 – 2410M @ 2.30 GHz
- Оперативная память: 4.00 ГБ
- Язык программирования: C++
- Компилятор: Microsoft Visual Studio 2012

Полученные результаты

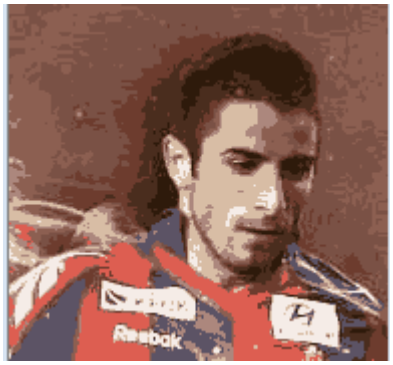
Разработанная реализация k-средних

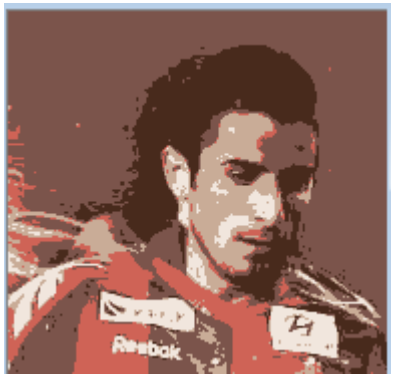
K = 2

K = 5

K = 10

Исходное изображение



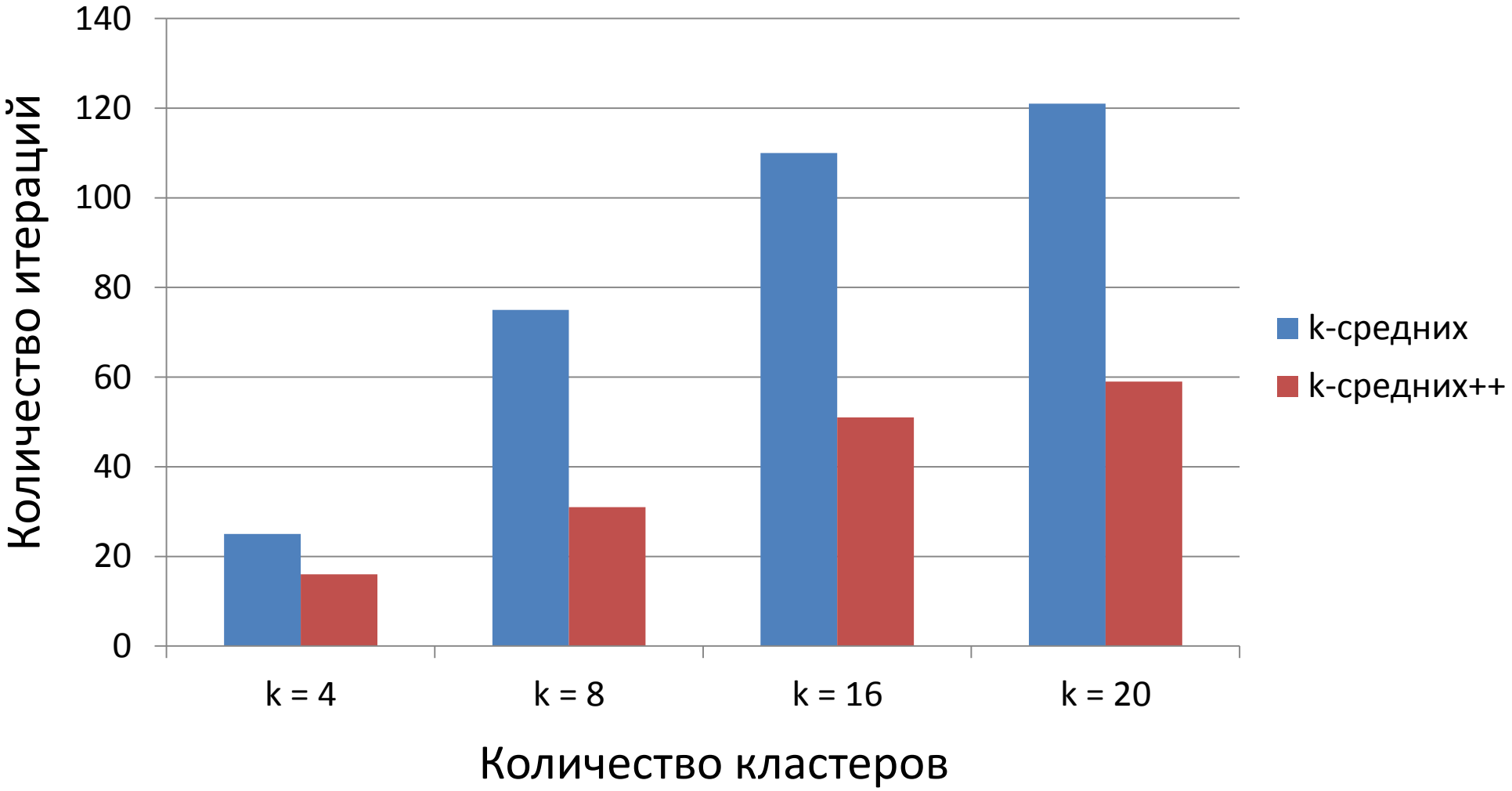


Реализация из OpenCV

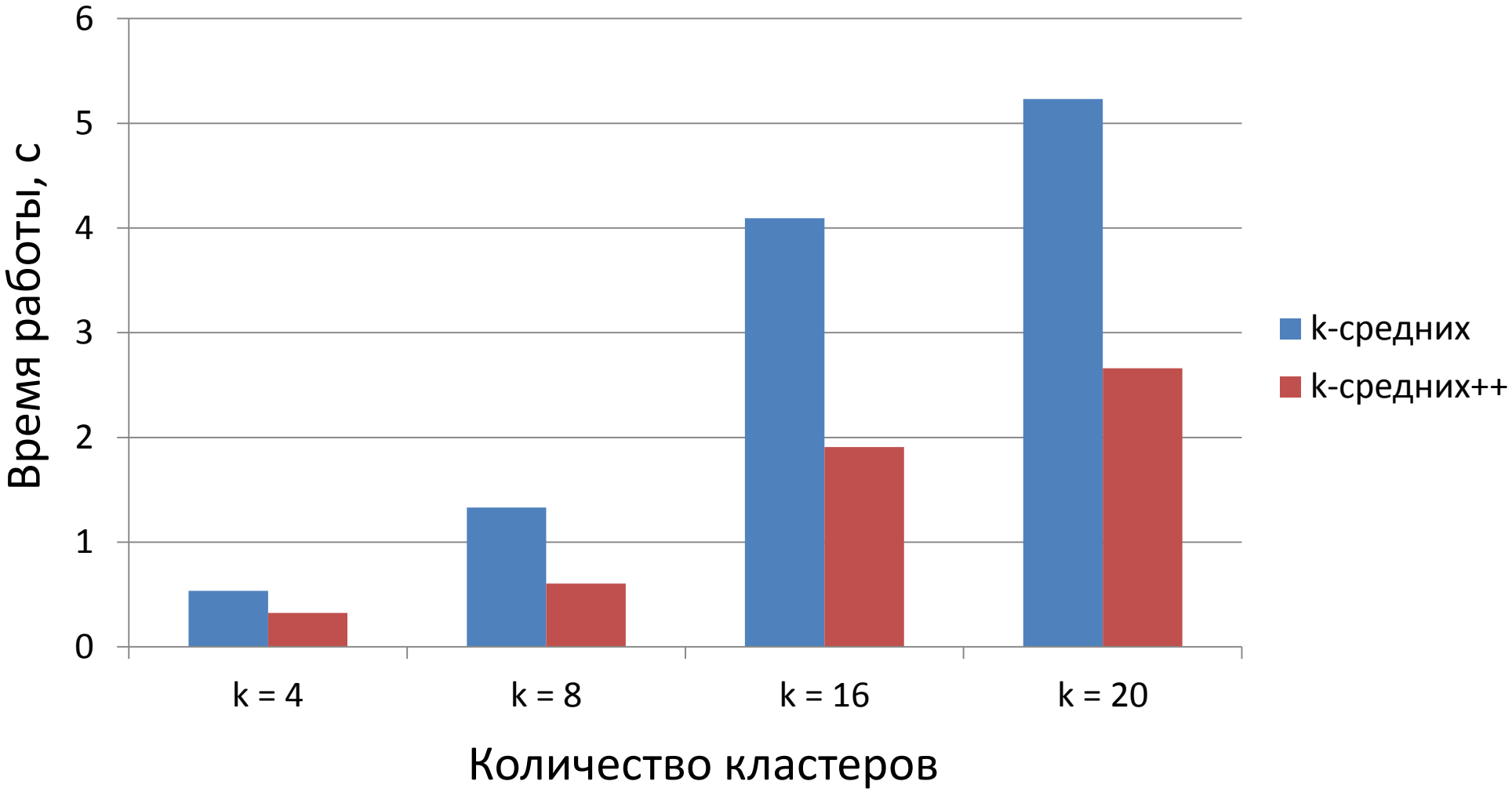
Сравнение k-средних и k-средних++

- Разрешение: 800 x 450 пикселей

Сравнение k-средних и k-средних++



Сравнение k-средних и k-средних++



МЕТОД ВОДОРАЗДЕЛА

Постановка задачи

- ***Входные данные:***

$$I = \big(I(x,y)\big), 0 \le I(x,y) \le 255,$$

$0 \le x < w, 0 \le y < h$ изображение в оттенках серого, w – ширина, h – высота.

- ***Выходные данные:***

$$M = \big(M(x,y)\big),$$

$0 \le M(x,y) < k, 0 \le x < w, 0 \le y < h$, причем k изначально неизвестно.

Основная идея

- I можно рассматривать как ***рельефную карту***: чем больше интенсивность, тем выше объект.
- Локальные минимумы начнём заливать водой.
- Спустя время вода из разных впадин может слиться, в таком случае на месте слияния строится дамба.

В результате получим сегменты – области, ограниченные дамбами.

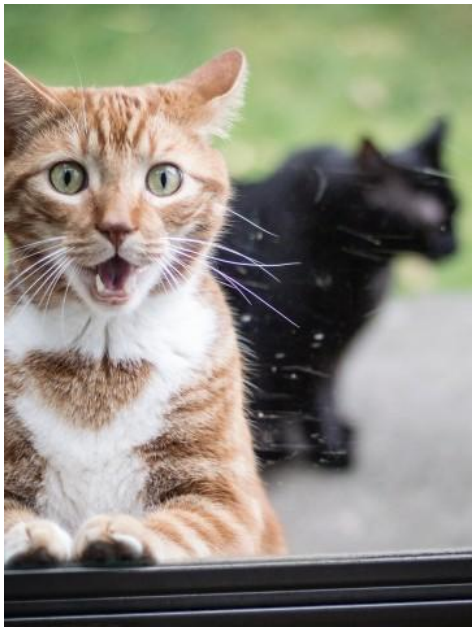
Схема алгоритма

Границы объектов = “холмы” на изображении.

1. Вычисление градиента в каждом пикселе.
2. Определение локальных минимумов.
3. Обход пикселей, применение приоритетной очереди (приоритет тем выше – чем меньше ключ, ключ - интенсивность).

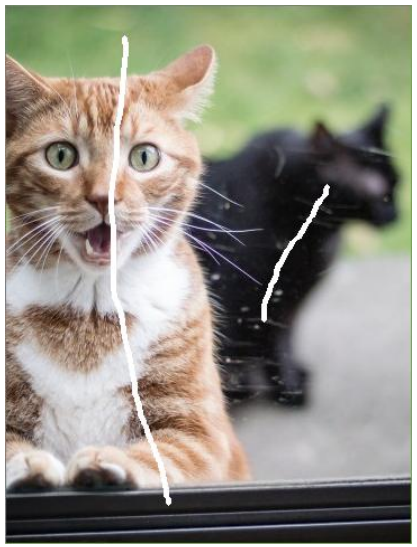
Недостатки алгоритма

Чтобы устранить чрезмерную сегментацию, разработан алгоритм с маркерами.



Алгоритм, основанный на маркерах

1. Обозначить на изображении маркером фон и интересующий объект.
2. Применить алгоритм водораздела, считая маркированные пиксели локальными минимумами.

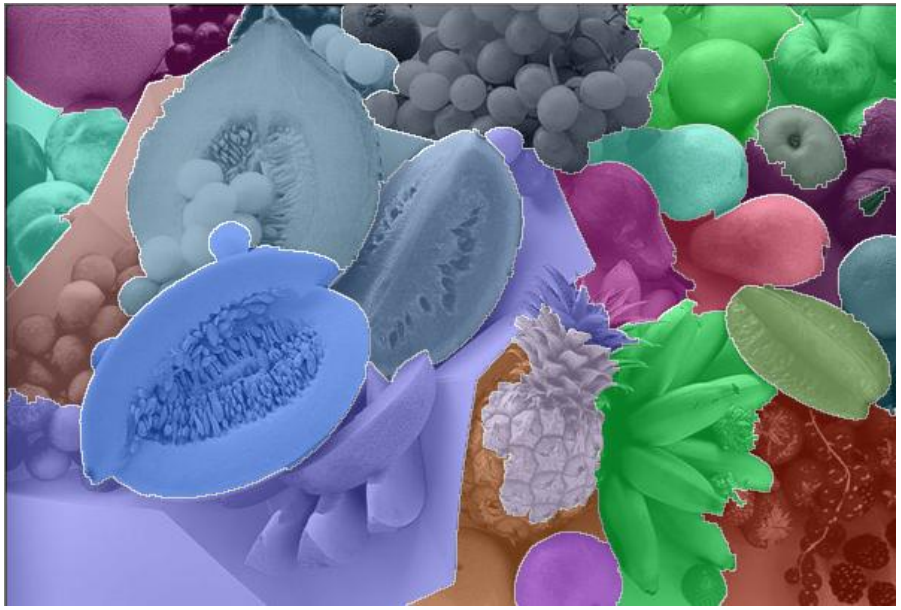


Тестовая инфраструктура

- ОС: Windows 8 x64
- Процессор: Intel® Core™ i5 – 2430M @ 2.4 GHz
- Оперативная память: 4.00 ГБ
- Язык программирования: C++
- Компилятор: Microsoft Visual Studio 2013

Вычислительные эксперименты (сравнение с алгоритмом в OpenCV)

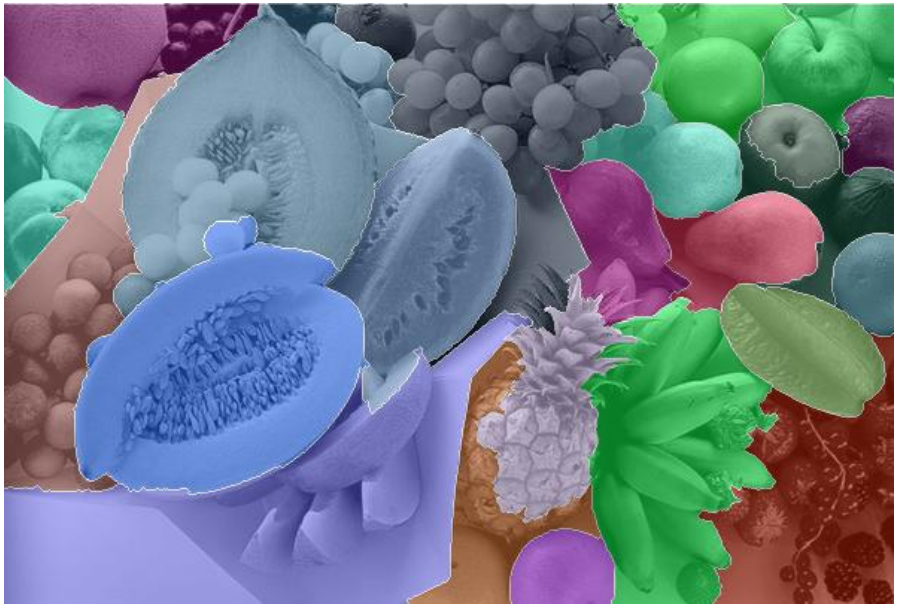
Разработанная реализация



Время работы: 4649.865 мс

Совпадение: 94,9%

OpenCV



75.495 мс

МЕТОД РАЗРЕЗА ГРАФА

Постановка задачи

- Входные данные:
  - Цветное изображение:

$$I = (I(x,y))_{0\le x\le w,\ 0\le y\le h},\ I(x,y) = \big(r(x,y), g(x,y), b(x,y)\big)$$
$$r(x,y), g(x,y), b(x,y) \in \{0, 1, \dots, 255\}$$

  - Маска:

$$M = (M(x,y))_{0\le x\le w,\ 0\le y\le h},\ M(x,y) \in \{0, 1, 2, 3\}$$

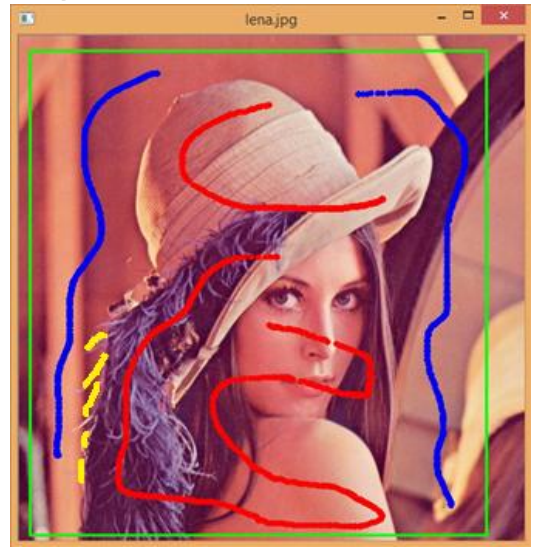
- Выходные данные:
  - Маска: $M = (M(x,y))_{0\le x\le w,\ 0\le y\le h},\ M(x,y) \in \{0, 1\}$

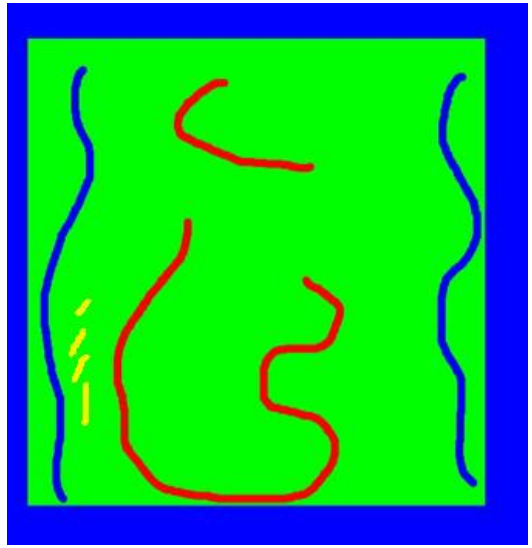
Схема алгоритма

1. Инициализация маски изображения
2. Создание и обучение моделей распределения интенсивностей фона и объекта
3. Построение графа специального вида для множества пикселей изображения
4. Обновление маски изображения
5. Повторять шаги 2-4, пока не будет достигнут желаемый результат

Описание алгоритма (1)

1. Инициализация маски изображения: выбор прямоугольной области для сегментации и маркировка пикселей, принадлежащих объекту/фону





   - Foreground ← все пиксели объекта(1) и возможно принадлежащие объекту пиксели(4)
   - Background ← все пиксели фона(0) и возможно принадлежащие фону пиксели(2)

Описание алгоритма (2)

2. Создание и обучение моделей распределения интенсивностей фона и объекта

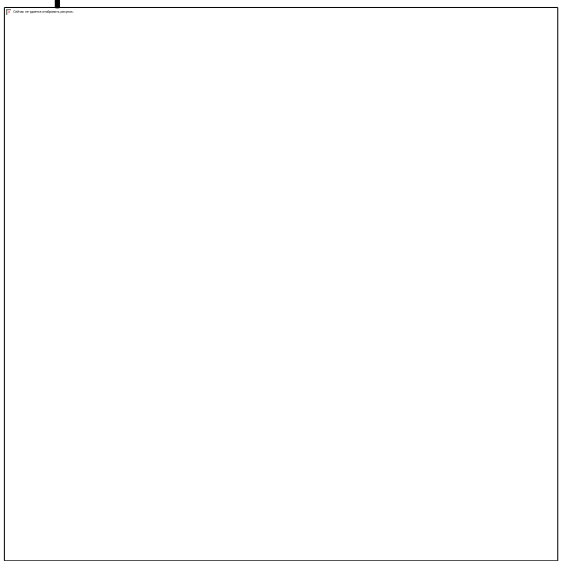
- Создание двух моделей распределения интенсивностей
- Каждая модель кластеризируется
- Назначение каждому пикселю изображения индекса наиболее подходящей компоненты

Описание алгоритма (3)

3. Построение графа специального вида для множества пикселей изображения
   - Вершины – все пиксели графа и две терминальные вершины
   - Ребра – связи между соседними пикселями, и каждый пиксель с терминальными вершинами
   - Вычисляются веса ребер между соседними пикселями, используя изменение градиента по 8-ми направлениям и нормальный закон распределения

$$N(m,n) = \frac{\gamma}{dist(m)} e^{-\beta \|z_m - z_n\|^2}, \text{ где}$$

$$\beta = \frac{1}{2\langle \|z_m - z_n\|^2 \rangle}$$



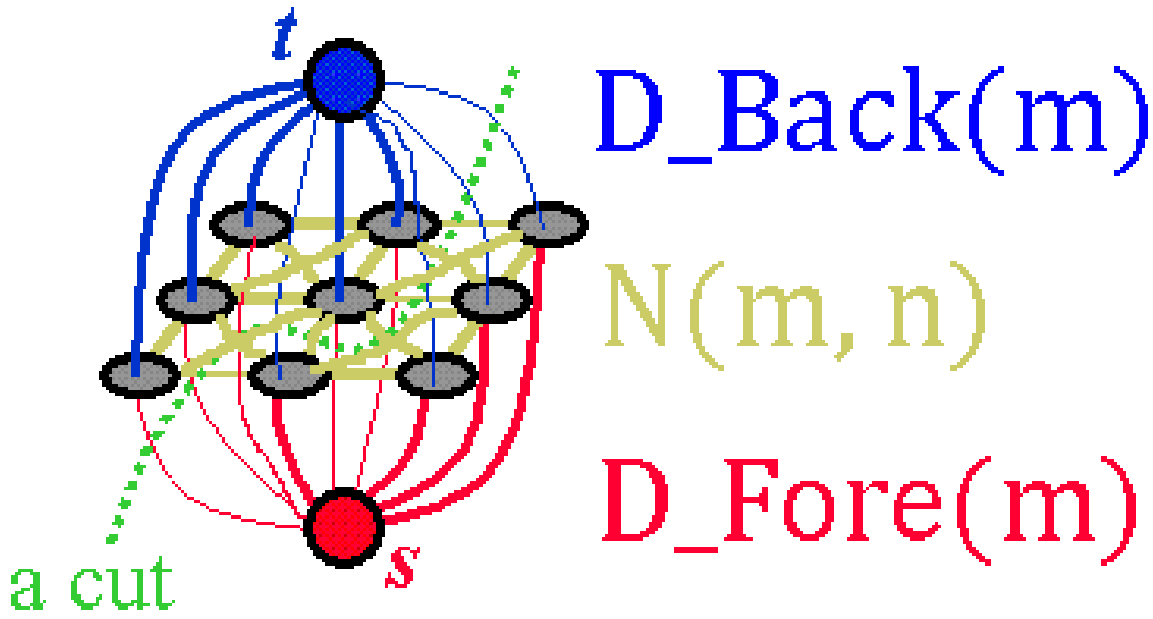
Описание алгоритма (4)

4. Обновление маски изображения
   - Вычисляются веса между терминальными вершинами и вершинами-пикселями изображения по следующей формуле:

$$D(m) = -log \sum_{i=1}^{K} \pi_i \frac{1}{\sqrt{det \sum_i}} e^{\left(-\frac{1}{2}[[z_m - \mu_i]]^T \sum_i^{-1} [z_m - \mu_i]\right)}$$

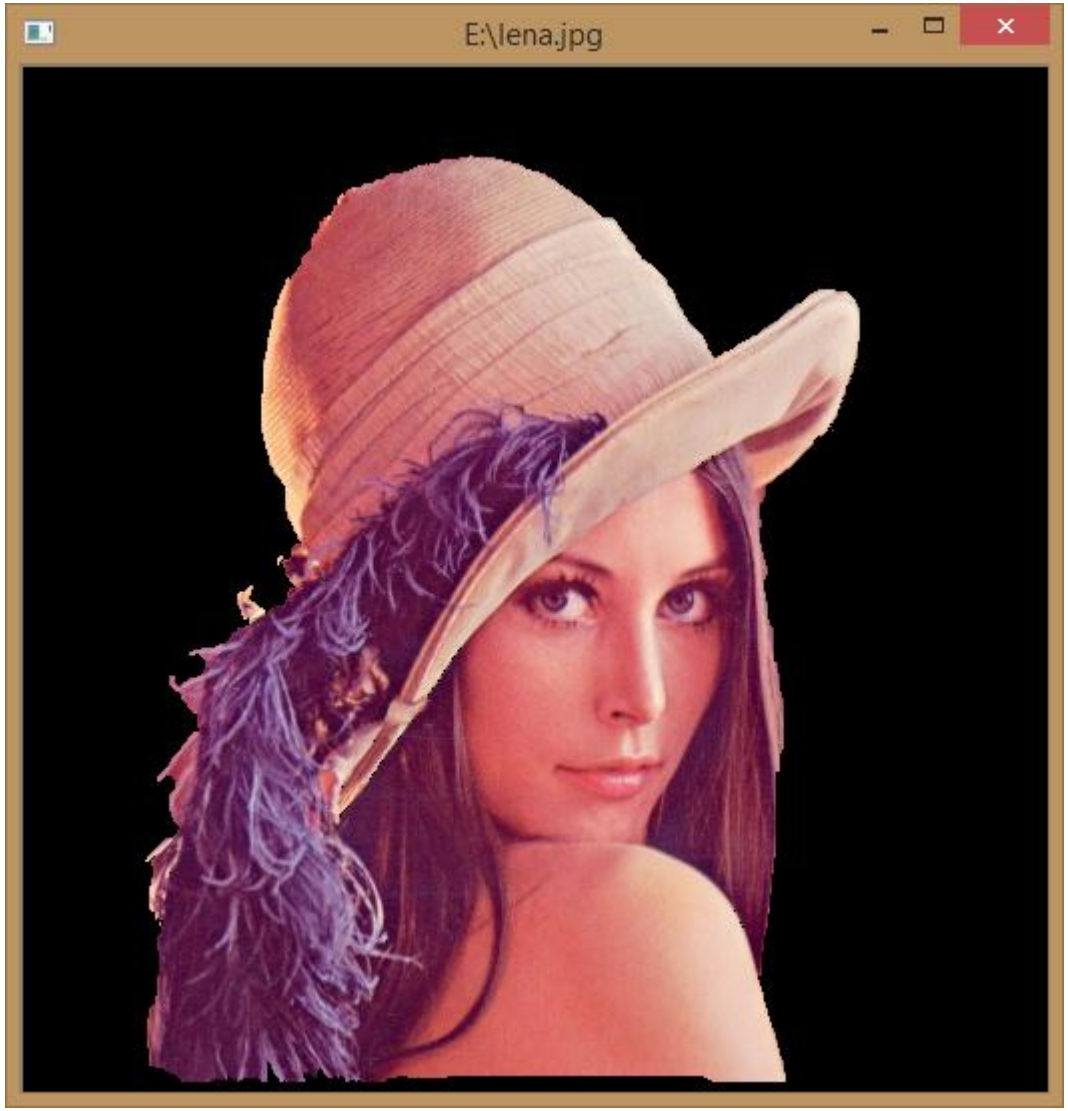
   - Выполняется разрез графа
     - Вершины, принадлежащие истоку -> пиксели объекта
     - Вершины, принадлежащие стоку -> пиксели фона

Описание алгоритма (5)



5. Повторять шаги 2-4 столько раз, сколько требуется для достижения желаемого результата

Результат



СЕГМЕНТАЦИЯ ВИДА «ФОН/НЕ ФОН» НА ВИДЕО ЧЕРЕЗ ВЫЧИТАНИЕ ФОНА

Постановка задачи

- ***Входные данные:***

Последовательность кадров

$$I = \{I_1, \dots, I_m\},$$

где $I_n = (I_n(x, y))_{0 \le x < w, 0 \le y < h}$,

$I_n(x, y) \in \{0,1, \dots, 255\}$,

- ***Выходные данные:***

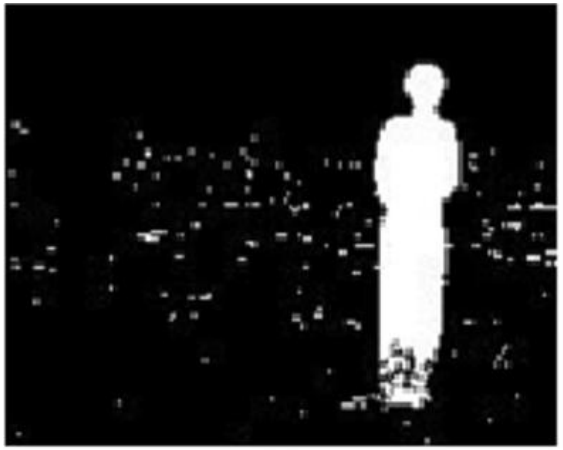
Последовательность бинарных масок

$$M = \{M_1, \dots, M_m\},$$

где $M_n = (M_n(x, y))_{0 \le x < w, 0 \le y < h}$,

$M_n(x, y) \in \{0, 1\}$,

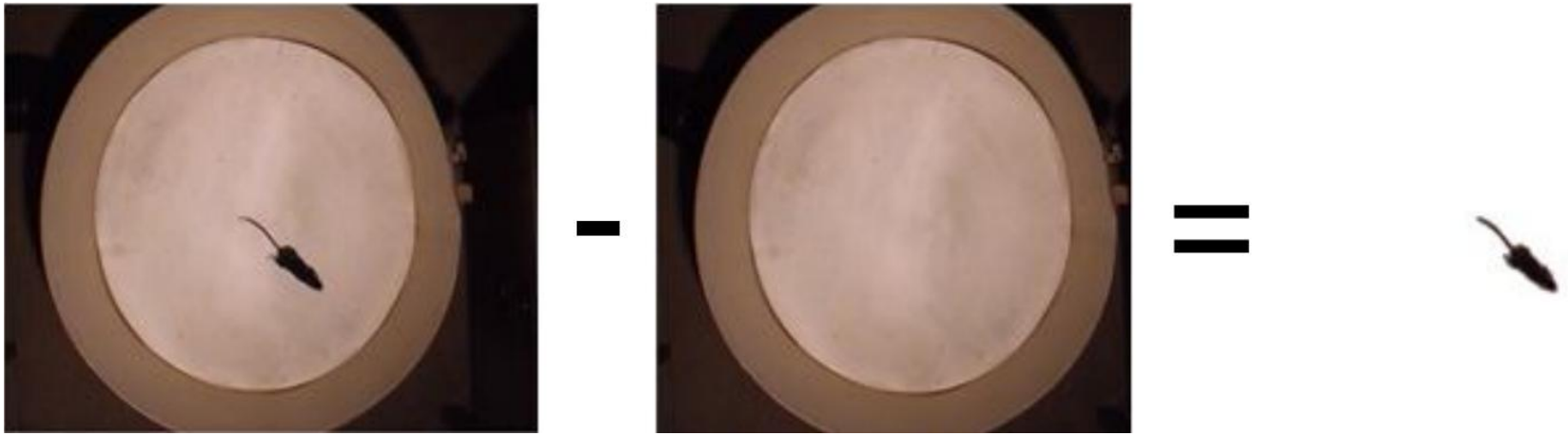
0 - фон, 1-интересующие объекты



Свойства задачи

- Камеру считают стационарной, не меняющей положение
- Фон считается неизменным или малоподвижным
- Объекты должны отличаться от фона

Простейший алгоритм вычитания фона



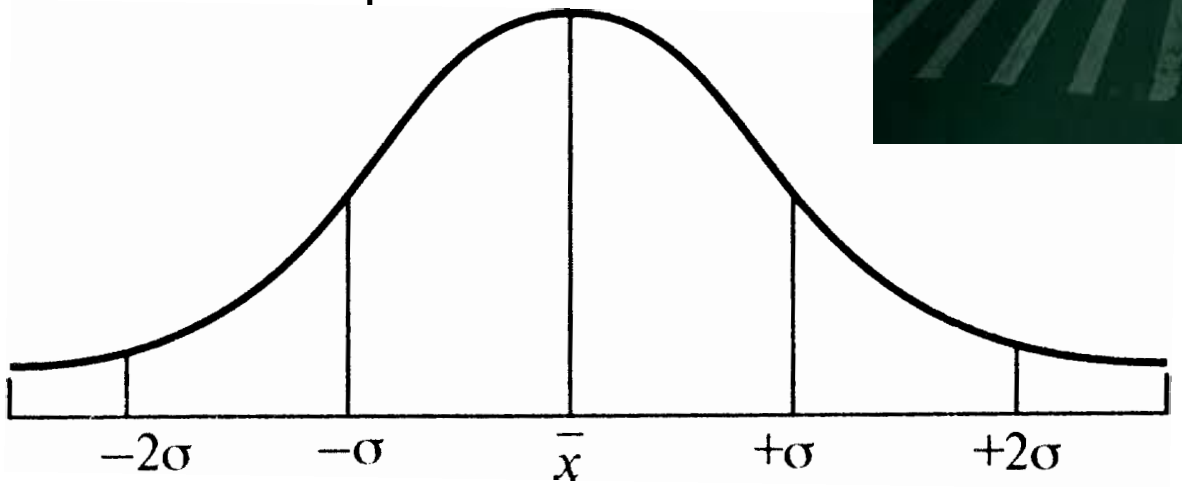
- Возьмем начальное изображение – «фон»
- Вычитаем фон из новых изображений с объектом
- Полученную разницу сравниваем с пороговым значением. Если разница больше порога, то пиксель принадлежит объекту

Недостатки данного метода

- Изменение фона (кто-то подвинул объект)
- Изменение освещения
- Что если нельзя найти чистое изображение фона?

Усовершенствование алгоритма

- Устранение недостатков
  - Шум камеры
  - Изменение освещенности



По «чистому» видео вычислим параметры для модели – нормального распределения

Моделирование фона

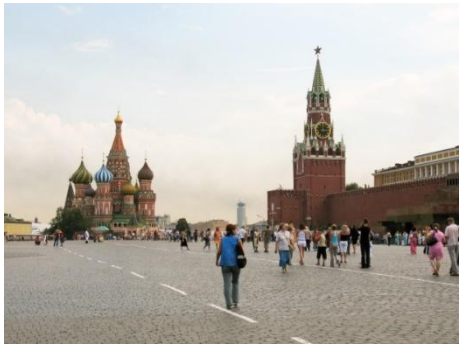
- Среднее значение: $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$
- Дисперсия: $\delta = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2$
- Вероятность X при вычисленных параметрах

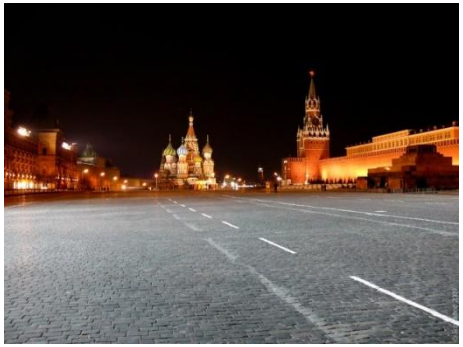
$$N\left(X_k \middle| \mu_j^k, \Sigma_j^k\right) = \frac{1}{(2\pi)^{\frac{D}{2}}\left|\Sigma_j^k\right|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(X_k-\mu_j^k\right)^T\left(\Sigma_j^k\right)^{-1}\left(X_k-\mu_j^k\right)}$$

- Если пиксель принадлежит $(\mu - 3\delta, \mu + 3\delta)$, пиксель принадлежит фону

Полученные результаты

Что делать если яркость меняется сильно?





- Обновление математического ожидания:

$$\mu_j^{k+1} = (1-\rho)\mu_j^k + \rho X_{k+1}$$

- Обновление дисперсии:

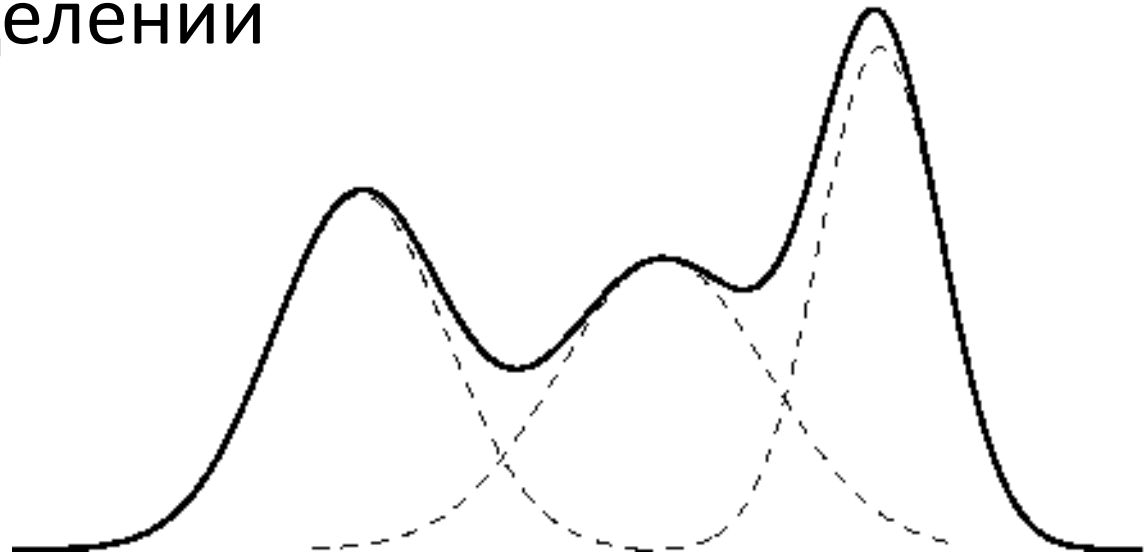
$$\left(\sigma_j^{k+1}\right)^2 = (1-\rho)\left(\sigma_j^k\right)^2 + \rho\left(X_{k+1} - \mu_j^{k+1}\right)\left(X_{k+1} - \mu_j^{k+1}\right)^T,$$

где $\rho = \alpha N\left(X_{k+1}\middle|\mu_j^k, \Sigma_j^k\right)$

Более сложный пример

Что делать, если искажения создаются разными объектами?

Представим модель фона, как смесь нормальных распределений



Смесь Гауссиан

Пусть K количество компонент в смеси.

Плотность вероятности значения пикселя X при модели из K нормальных распределений:

$$P(X_k) = \sum_{j=1}^{s} \omega_j^k N\left(X_k \middle| \mu_j^k, \Sigma_j^k\right)$$

- Инициализируем первую компоненту по первому кадру. Остальные компоненты остаются пустыми.
- Просматриваем следующие кадры. Сравниваем пиксели с каждой компонентой.
- Обновляем смесь

Модель фона

- Распределения сортируются в порядке уменьшения величины: $r_j^k = \frac{\omega_j^k}{\sigma_j^k}$.
- Моделью фона являются первые K распределений*, которые удовлетворяют правилу

$$B^k = argmin_b \left\{ \sum_{j=1}^{b} \omega_j^k > T \right\}$$

* Пиксель удовлетворяет распределению, если выполняется неравенство (расстояние Махаланобиса)

$$\sqrt{\left(X_{k+1} - \mu_j^k\right)^T \left(\Sigma_j^k\right)^{-1} \left(X_{k+1} - \mu_j^k\right)} < 2{,}5\sigma_j^k$$

Обновление модели

- Если новое значение пикселя удовлетворяет хотя бы одному распределению
  - обновляем мат. ожидание и дисперсию тех компонент, которым удовлетворяет новое значение
  - обновляем веса:

$$\omega_j^{k+1} = \begin{cases} (1-\alpha)\omega_j^k + \alpha \text{ , если соответствие найдено} \\ (1-\alpha)\omega_j^k \quad \text{, иначе} \end{cases}$$

- Если соответствие не найдено, то крайнее в смысле введенного отношения порядка распределение замещается новым.

Пример

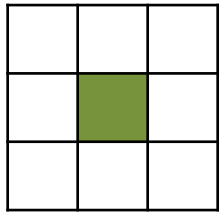
(1) (2) (3) (4)

1. Исходный кадр
2. Модель фона
3. Мат.ожидание второй компоненты
4. Бинарная маска

Visual Background Extractor (ViBe) Описание алгоритма (1)

- Каждому пикселю соответствует модель из K различных значений

  $P = \{P_1, P_2 \dots, P_k\}$.

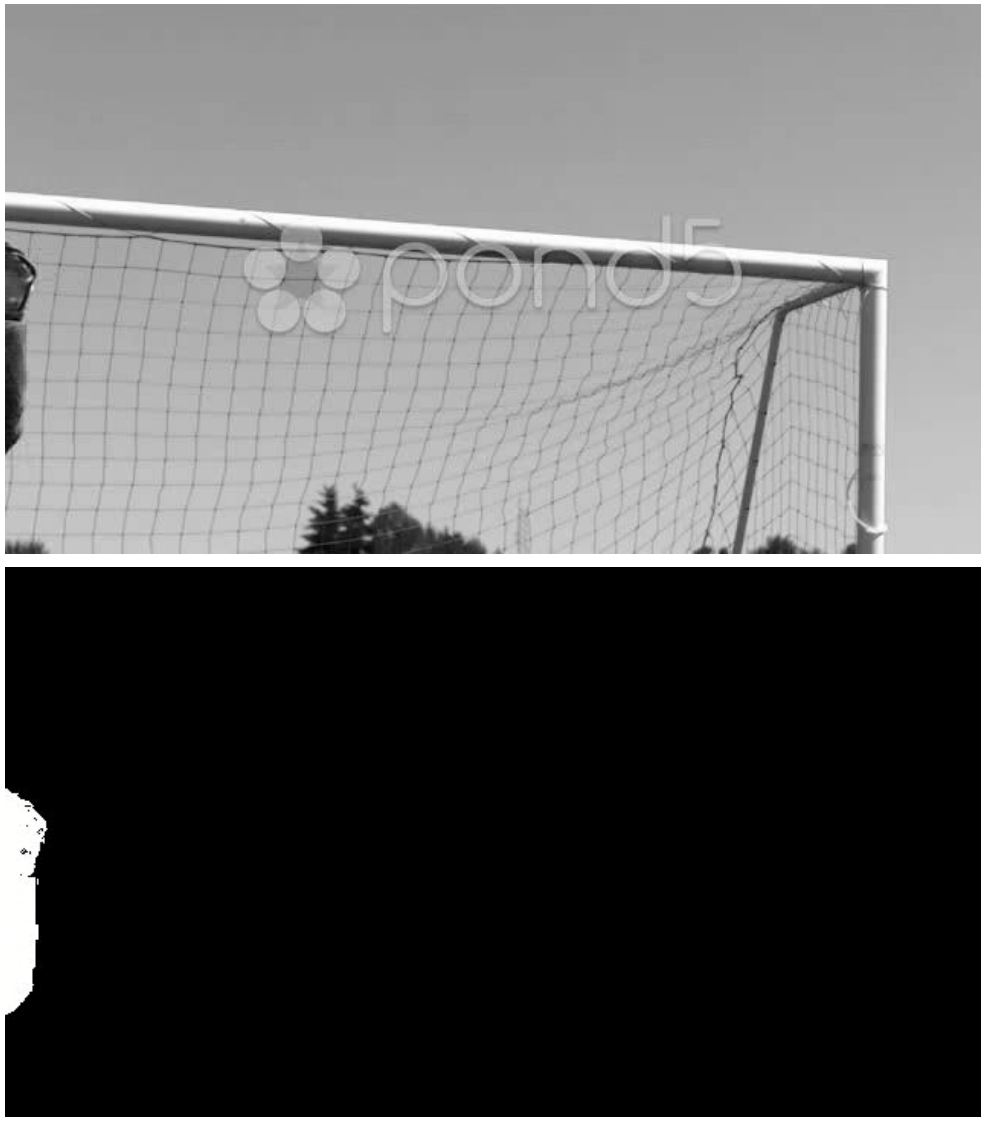


- Начальная инициализация
  - Модель каждого пикселя заполняется значениями случайных пикселей из окрестности размером 3x3.

Visual Background Extractor (ViBe) Описание алгоритма (2)

- Вычитание фона
  - Сравниваем новое значение пикселя со всеми значениями в модели фона для данного пикселя.
  - Если количество совпадений больше некоторого значения, то пиксель считается фоновым.
- Обновление модели
  - Если пиксель принадлежит фону, то заменяем одно значение в модели данного пикселя и одного из соседних на новое.

Пример работы простейшего алгоритма вычитания фона



Пример работы алгоритма, основанного на GMM

Пример работы ViBe

Тестовая инфраструктура

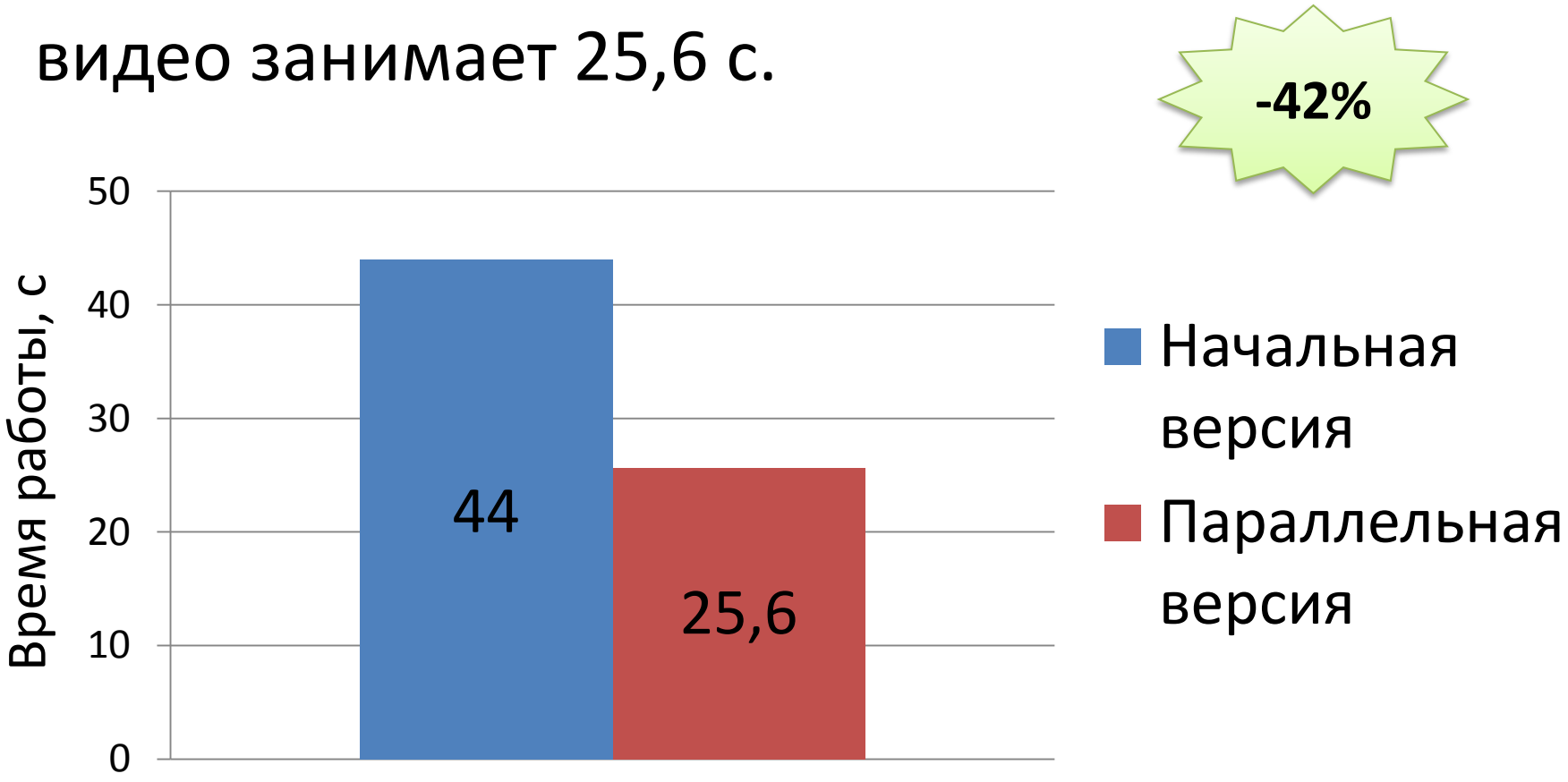
- ОС: Windows 8
- Процессор: Intel Core i5 3210M 2 ядра, 2.9 GHz
- Оперативная память: 6ГБ DDR3
- Язык программирования: C++
- Компилятор, отладчик: Microsoft Visual Studio 2010

Оптимизация алгоритма, основанного на GMM

Обработка видео в оттенках серого (1280x720) продолжительностью 59с начальной версией программы занимает 44с. С помощью библиотеки OpenCV видео обрабатывается 9,8 с.

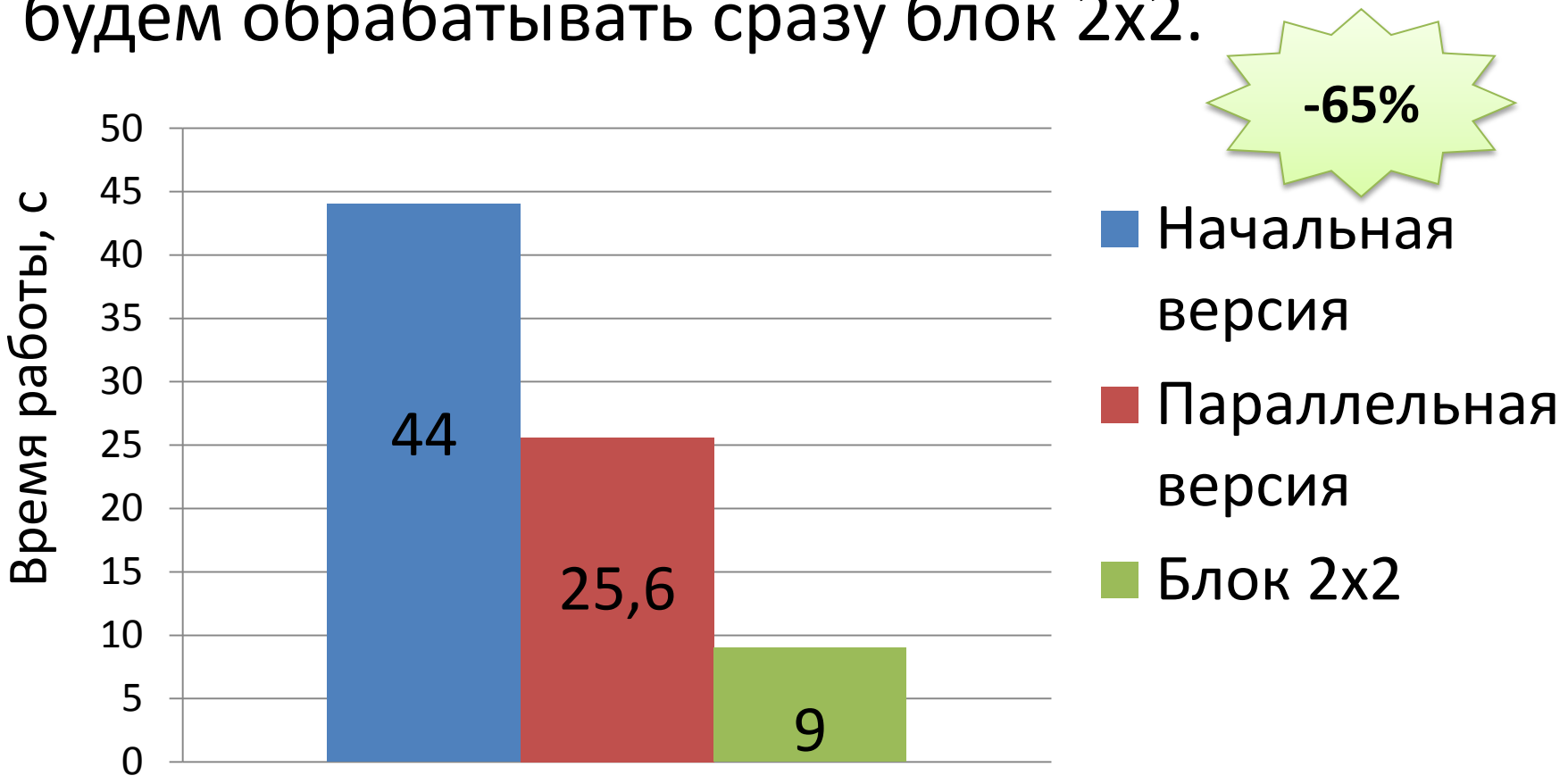
Оптимизация алгоритма, основанного на GMM

При параллельной обработке столбцов обработка видео занимает 25,6 с.



Оптимизация алгоритма, основанного на GMM

Считая, что объекты на видео достаточно велики, будем обрабатывать сразу блок 2x2.



Вопросы

- ???