Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

СЫКТЫВКАРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт точных наук и информационных технологий

Кафедра прикладной математики

Допустить к защите

Зав. кафедрой прикладной математики, д-р тех. наук

_____ В.Л. Никитенков

"____" _____ 2012 г.

Выпускная квалификационная работа

Моделирование фона видеоизображения методом смеси Гауссианов.

Научный руководидель _____В.Л. Никитенков

д.т.н., профессор

Исполнитель, студент _____А.Е. Ануфриев

155 гр.

Аннотация

В данной работе рассматривается моделирование фона видеоизображения смесью гауссовых распределений, построение модели фона, ускорение алгоритма расчёта модели фона.

Ключевые слова: моделирование фона, смесь гауссиан, EM алгоритм, k-means.

In this paper the simulation of the background video with a mixture of Gaussian (MOG), construction of the model background, the acceleration of the algorithm for calculating the background model.

Keywords: modeling the background, a mixture of Gaussian, EM algorithm, k-means.

Содержание

1. Введение.

В задачах распознавания образов есть класс задач, в которых требуется определить движущиеся объекты на видео последовательности для дальнейшего определения к какому классу принадлежит объект, подсчёта объектов или расчёта скорости и траектории движения объекта. Можно вести поиск интересующего нас объекта в каждом кадре видео по всем его пикселям, но для облегчения задачи можно определять сразу относится ли пиксель к фону, и тогда его можно не рассматривать в расчётах. Для решения задачи разделения каждого пикселя видео на фон и не фон, можно применить несколько способов:

1) Вычитать из текущего кадра предыдущий и оценивать по некоторому порогу относится пиксель к фону, либо нет. Данный способ будет давать большую погрешность, когда фон не является статичным (например: падающий снег, колышущиеся деревья и трава, пузырьки в воде рис. 1) либо камера сильно "шумит".

2) Строить некоторую статистическую модель для фона, в которой мы можем учитывать изменения фона по времени и шум камеры.

В данной работе рассматривается моделирование фона видео изображения смесью гауссиан при помощи EM и k-means алгоритмов, построение модели фона, ускорение стандартных алгоритмов вычисления фона.

Данная работа актуальна для автоматизации учёта, робототехники и систем безопасности.

Рис. 1 Пример нестатичного фона

2. Реализация

Рассмотрим формулы для реализации данного алгоритма.

2.1. Построение функции правдоподобия. EM алгоритм.

Для набора данных (пикселей) будем искать функцию правдоподобия в виде суммы гауссиан с помощью EM - алгоритма (expectation - maximization - algorithm)[2],[3],[4]. В общем случае идея алгоритма заключается в следующем.

Пусть плотность распределения на множестве **X** имеет вид смеси **k** распределений:

$$p(\mathbf{x}) = \sum_{j=1}^{k} \omega_j \, p_j(\mathbf{x}), \sum_{j=1}^{k} \omega_j = 1, \omega_j \geq 0.$$

где p_j - функция правдоподобия j-й компоненты смеси, ω_j - ее априорная вероятность. Пусть функции правдоподобия принадлежат параметрическому семейству распределений

$$\phi(\mathbf{x},\mathrm{H})$$

и отличаются только значениями параметра

$$p_j = \phi(\mathbf{x}; \mathrm{H}_j)$$

Задача разделения смеси заключается в том, чтобы, имея выборку $\mathbf{X}^m$ случайных и независимых наблюдений из смеси $p(\mathbf{x})$, зная число k и функцию ϕ, оценить вектор параметров $\Theta = (\omega_{1,\ldots,}\omega^{k;}\mathrm{H}_{1,\ldots,}\mathrm{H}_k)$

Искусственно вводится вспомогательный вектор скрытых переменных G, обладающий двумя замечательными свойствами.

а) С одной стороны, он может быть вычислен, если известен вектор вектор параметров Θ

б) С другой стороны, поиск максимума правдоподобия сильно упрощается, если известны значения скрытых переменных.

EM - алгоритм состоит из итерационного повторения двух шагов:

На **E – шаге** вычисляется ожидаемое значение (expectation) вектора скрытых переменных G по текущему приближению вектора параметров Θ. Обозначим через $p(\mathbf{x}, \mathrm{H}_j)$ плотность вероятности того, что объект **x** получен из j-й компоненты смеси. По формуле условной вероятности

$$p(\mathbf{x},\mathrm{H}_j) = p(\mathbf{x}) \, P(\mathrm{H}_j|\mathbf{x}) = \omega_j \, p_j(\mathbf{x})$$

Введём обозначение

$$g_{ij} = P(\mathrm{H}_j|\mathbf{x}_i)$$

Это неизвестная апостериорная вероятность того, что обучающий объект $\mathbf{x}_i$ получен из j-й компоненты смеси. Возьмём эти величины в качестве скрытых переменных.

$$\sum_{j=1}^{k} g_{ij} = 1,$$

для любого $i = 1..m$, так как имеет смысл полной вероятности принадлежать объекту $\mathbf{x}_i$ одной из **k** компонент смеси. Из формулы Байеса:

$$g_{ij} = \frac{\omega_j \, p_j(x_i)}{\sum_{s=1}^{k} \omega_s \, p_s(x_i)}$$

На **M – шаге** решается задача максимизации правдоподобия (maximization) и находится следующее приближение вектора Θ по текущим значениям векторов G и Θ. Будем максимизировать логарифм полного правдоподобия:

$$Q(\Theta) = \ln \prod_{i=1}^{m} p(x_i) = \sum_{i=1}^{m} \sum_{j=1}^{k} \omega_j \, p_j(x_i)$$

Решая оптимизационную задачу Лагранжа с ограничением на сумму ω_j, находим [3]:

$$\omega_j = \frac{1}{m} \sum_{i=1}^{m} g_{ij}, j = 1..k$$

$$\mathrm{H}_j = argmax_{\mathrm{H}} \sum_{\substack{i=1 \\ m}} g_{ij} \ln \phi(x, \mathrm{H}), \; j = 1..k$$

Рассмотрим данный алгоритм более подробно для смеси гауссовых распределений (mixture of gaussian):

Формула функции плотности многомерного гауссовского распределения имеет вид:

$$\eta(\mathbf{x} \mid \mu, \Sigma) = \frac{1}{(2\pi)^{M/2} \left|\Sigma\right|^{1/2}} \, e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \, \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})}. \qquad (1)$$

Здесь:

x вектор $(x_1, x_2, x_3, \ldots, x_M)$, размерности M;

Σ ковариационная матрица, а $|\Sigma|$ — её детерминант;

$\boldsymbol{\mu}$ вектор средних значений $\mathbf{x}$.

Обозначение ‘‘$\eta(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma)$’’ указывает на то, что функция получает вектор $\mathbf{x}$, как аргумент, и параметризуется значениями Σ и $\boldsymbol{\mu}$.

Плотность вероятности для смеси из K гауссовских распределений, вес каждой из которых задан ω_k, задаётся так:

$$f_{\mathbf{x}}(\mathbf{x}) = \sum_{k=1}^{K} \omega_k \, \eta(\mathbf{x} \mid \boldsymbol{\mu}_k, \Sigma_k). \tag{2}$$

Так как функция $f_{\mathbf{x}}(\mathbf{x})$ есть плотность вероятности, её интеграл должен быть равен единице, очевидно, что сумма всех весов $\sum \omega$ тоже должна равняться единице, и все $\omega_k > 0$.

Функция правдоподобия рассчитывается всегда для конечного набора значений $\mathbf{x}_n$, как произведение их плотностей вероятности:

$$L = f_{\mathbf{x}}(\mathbf{x}_1) f_{\mathbf{x}}(\mathbf{x}_2) \cdots f_{\mathbf{x}}(\mathbf{x}_N) = \prod_{n=1}^{N} f_{\mathbf{x}}(\mathbf{x}_n).$$

Функция правдоподобия для смеси из K гауссовских распределений для всего набора значений $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ имеет вид:

$$L = \prod_{n=1}^{N} \sum_{k=1}^{K} \omega_k \eta(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k). \tag{3}$$

В практическом смысле, в ходе выполнения какого-то алгоритма, нас чаще всего интересует как изменялась функция правдоподобности, а не её абсолютное значение. Поэтому вместо L можно взять её логарифм, при этом $\log L$ превратит произведение в сумму:

$$\log L = \log \left[\prod_{n=1}^{N} \sum_{k=1}^{K} \omega_k \eta(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k) \right]$$

$$= \log \sum_{k=1}^{K} \omega_k \eta(\mathbf{x}_1 \mid \boldsymbol{\mu}_k, \Sigma_k) + \cdots + \log \sum_{k=1}^{K} \omega_k \eta(\mathbf{x}_N \mid \boldsymbol{\mu}_k, \Sigma_k)$$

$$= \sum_{n=1}^{N} \log \sum_{k=1}^{K} \omega_k \eta(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k).$$

Существует один важный практический аспект. При подсчёте сумм плотностей, часто значения плотностей оказываются настолько малыми, что в операциях с плавающей точкой происходит исчезновение порядка (floating point underflow). В арифметике

IEEE-754 исчезновение порядка приводит к тому что число денормализуется и плавно, а не резко, приближается к нулю. Большинство сопроцессоров операции с денормализованными числами выполняют через микропрограмму, а не аппаратно, что обычно медленнее на порядок. Потому целесообразно и при вычислении $\eta(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma)$ переходить к логарифмированию:

$$\log \eta(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = -\frac{M}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

$$= -\frac{1}{2}\left(M\log(2\pi) + \log|\Sigma| + (\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

После логарифмирования, значения должны быть восстановлены экспоненцированием.

2.2. Формула log – sum – exp

В формуле для смеси гауссовских распределений каждая компонента плотности имеет вес ω_k. Будем обозначать логарифм плотности распределения с весом ω как γ:

$$\gamma = \log \eta(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) + \log \omega$$

Формула для расчёта $\log L$ тогда примет вид:

$$\log L = \sum_{n=1}^{N} \log\left(\sum_{k=1}^{K} \exp(\gamma_k)\right).$$

Значение $exp(\gamma_k)$ может опять таки оказаться настолько малым, что это приведёт к исчезновению порядка. Для того чтобы избежать этого, при вычислении логарифма суммы сначала находится самое большое $\gamma_{\max}$, а затем применяется формула

$$\log\left(\sum_{k=1}^{K} \exp(\gamma_k)\right) = \gamma_{\max} + \log\left(\sum_{k=1}^{K} \exp(\gamma_k - \gamma_{\max})\right),$$

которая гарантирует, что хотя бы одно значение не вызовет исчезновения порядка, а остальные значения в таком случае всё равно не окажут влияния на сумму в силу своей малости. Такой трюк известен под названием «*формула log-sum-exp*».

2.3. Эффективное вычисление функции плотности

Функция плотности гауссовского распределения (1) содержит скаляр

$$(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) = \theta^T \Sigma^{-1}\theta.$$

Конечно, можно вычислить это значение, как говорится, «в лоб», найдя обратную ковариационную матрицу и выполнив две операции умножения «матрица-вектор». Тем не

менее, существует способ сделать это проще и компактнее, что не маловажно для систем, обладающих скромными ресурсами. Свойства ковариационной матрицы позволяют осуществить упрощение, потому что эта матрица *симметричная* и, в большинстве практических случаев, *положительно-определённая*, а значит её можно факторизовать, пользуясь *разложением Холецкого*, на две треугольные матрицы

$$\Sigma = \mathbf{C}\mathbf{C}^T,$$

где:

- $\mathbf{C}$ нижняя треугольная матрица со строго положительными диагональными элементами;
- $\mathbf{C}^T$ верхняя треугольная матрица.

Любую симметричную положительно-определённую матрицу можно представить в таком виде. Взяв нижнюю треугольную матрицу $\mathbf{C}$, мы можем очень быстро, через прямую подстановку, найти вектор u такой, что будет выполняться равенство

$$\mathbf{C}u = \theta.$$

Если теперь заменить вектор θ на $\mathbf{C}u$, получим:

$$\theta^T \Sigma^{-1} \theta = (\mathbf{C}u)^T (\mathbf{C}\mathbf{C}^T)^{-1} \mathbf{C}u$$

$$= u^T \mathbf{C}^T \mathbf{C}^{-T} \mathbf{C}^{-1} \mathbf{C}u$$

$$= u^T u = u \cdot u.$$

Алгоритм разложения Холецкого достаточно прост и эффективен в реализации на ЭВМ, и, в том случае если размерность вектора $\mathbf{x}$ невелика и известна заранее, все циклы алгоритма, вычисляющие коэффициенты матриц разложения, можно развернуть.

2.4. Практическая реализация EM алгоритма.

Рассмотрим реализацию EM алгоритма для моделирования фона.

2.4.1. Условия задачи

Дано N точек $(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N)$ размерности M. Известно, что все точки принадлежат K многомерным гауссовским распределениям вида $\omega_k \, \eta(\mathbf{x} \mid \boldsymbol{\mu}_k, \Sigma_k)$ с неизвестными параметрами $\boldsymbol{\mu}_k$, Σ_k, ω_k:

- $\boldsymbol{\mu}_k$ вектор средних значений для k-го распределения;

Σ_k ковариационная матрица $M \times M$ для k-го распределения;

ω_k вес k-го распределения, $\sum_k \omega_k = 1$.

Каждая точка $\mathbf{x}_n$ имеет некоторую вероятность присутствия в распределении k. Эту вероятность мы будем обозначать как P_{nk}. Все параметры P_{nk} для всех точек удобно свести в матрицу $P_{N\times K}$:

$$P_{N\times K} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \cdots & P_{1K} \\ P_{21} & P_{22} & P_{23} & \cdots & P_{2K} \\ & & & & \\ P_{N1} & P_{N2} & P_{N3} & \cdots & P_{NK} \end{bmatrix}.$$

Правдоподобность и вероятности P_{nk}

Вероятностную плотность для точки с координатами $\mathbf{x}$ будем обозначать как $P(\mathbf{x})$. Это вероятность попасть в точку по этим координатам, если случайным образом выбирать её из всей выборки. Правдоподобность параметров распределения будем обозначать как L. Наша задача максимизировать L, подобрав параметры $\boldsymbol{\mu}_k$, Σ_k, ω_k. Так как мы полагаем точки с данными независимыми, их L будет произведением вероятностей присутствия точки из выборки в $\mathbf{x}_n$:

$$L = \prod_{n=1}^{N} P(\mathbf{x}_n) = P(\mathbf{x}_1) \times P(\mathbf{x}_2) \times \cdots \times P(\mathbf{x}_N).$$

Для того, чтобы посчитать L, нам нужно найти плотности распределения в точках $\mathbf{x}_n$ и перемножить их. Для каждой точки $\mathbf{x}_n$ плотность определяется как сумма плотностей из K распределений, пропорционально весу k-го распределения:

$$P(\mathbf{x}_n) = \sum_{k=1}^{K} \omega_k \, p_j \, (\mathbf{x}).$$

Так как значения P_{nk}, то есть значения строки n матрицы $P_{N\times K}$, есть вероятности вхождения $\mathbf{x}_n$ в распределение k, их сумма должна равняться единице:

$$\sum_{k=1}^{k} P_{nk} = 1.$$

Значит $P(\mathbf{x}_n)$ это сумма

$$P(\mathbf{x}_n) = P_{n1} \cdot P(\mathbf{x}_n) + P_{n2} \cdot P(\mathbf{x}_n) + \cdots + P_{NK} \cdot P(\mathbf{x}_n),$$

но нам также известно, что

$$P(\mathbf{x}_n) = \omega_1 \, \eta(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \Sigma_1) + \omega_2 \, \eta(\mathbf{x}_n \mid \boldsymbol{\mu}_2, \Sigma_2) + \cdots + \omega_K \, \eta(\mathbf{x}_n \mid \boldsymbol{\mu}_K, \Sigma_K).$$

Последнее означает, что каждое значение P_{nk} можно вычислить по формуле:

$$P_{nk} = \frac{\omega_k \, \eta(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)}{P(\mathbf{x}_n)}.$$

При заполнении матрицы $P_{N\times K}$ значениями P_{nk}, вычисления рационально построить по следующей схеме:

Для каждой строки n матрицы $P_{N\times K}$:

а) Заполнить строку значениями $P_{nk} \cdot P(\mathbf{x}_n) = \omega_k \, \eta(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \Sigma_k)$.

б) Подсчитать сумму всех элементов в строке $\sum P_{nk}$.

в) Поделить все элементы строки на сумму, полученную на предыдущем шаге.

2.4.2. Формулы максимизации

Значения $\bar{\boldsymbol{\mu}}_k$, $\bar{\Sigma}_k$, $\bar{\omega}_k$ на шаге максимизации вычисляются по следующим формулам:

$$\bar{\omega}_k = \frac{1}{N} \sum_n P_{nk} \tag{1}$$

$$\bar{\boldsymbol{\mu}}_k = \frac{\sum_n P_{nk} \cdot \mathbf{x}_n}{\sum_n P_{nk}} \tag{2}$$

$$\bar{\Sigma}_k = \frac{\sum_n P_{nk} \cdot (\mathbf{x}_n - \bar{\boldsymbol{\mu}}_k)(\mathbf{x}_n - \bar{\boldsymbol{\mu}}_k)^T}{\sum_n P_{nk}} \tag{3}$$

Пошаговое выполнение алгоритма

Пользователь предоставляет данные $\mathbf{x}_n$, размерности N, M и K, значение ε и, возможно, вектор начальных значений для $\boldsymbol{\mu}_k$. Реализация алгоритма может состоять из следующих шагов:

1. На первом шаге:

   а) Установить веса ω_k, как $\omega_k = \frac{1}{k}$.

   б) Установить Σ_k в $I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ & & & \\ 0 & 0 & \cdots & 1 \end{bmatrix}$.

   в) Выбрать $\boldsymbol{\mu}_k$, как K случайно взятых значения из набора $\mathbf{x}_n$, или же взять начальные значения, предоставленные пользователем.

2. На втором шаге:
   а) Выполнить шаг *E*:
      а.1) вычислить P_{nk} и $P(\mathbf{x}_n)$;
      а.2) вычислить сумму всех логарифмов $P(\mathbf{x}_n)$;
      а.3) сохранить $\sum_n \log P(\mathbf{x}_n)$ в переменную *loglike_prev*.
   б) Выполнить шаг *M*:
      б.1) пользуясь формулой (1) и весами $P_{N\times K}$, вычислить $\bar{\omega}_k$;
      б.2) пользуясь формулой (2) и весами $P_{N\times K}$, вычислить $\bar{\boldsymbol{\mu}}_k$;
      б.3) пользуясь формулой (3) и весами $P_{N\times K}$, вычислить $\bar{\Sigma}_k$.
3. На третьем шаге:
   а) Выполнить шаг *E*, как в 2а, но сохраняя результат 2а.3 в переменную *loglike*.
   б) Вычислить $abs(loglike - loglike\_prev)$, и сравнить с ε, заданным пользователем. Если $abs(loglike - loglike\_prev) < \varepsilon$, тогда перейти на шаг 4.
   в) Выполнить шаг *M*, как в 2б.
   г) Перейти на шаг 3.
4. Завершить выполнение алгоритма с успешным статусом, а значения $\bar{\boldsymbol{\mu}}_k$, $\bar{\Sigma}_k$, $\bar{\omega}_k$ вернуть пользователю.

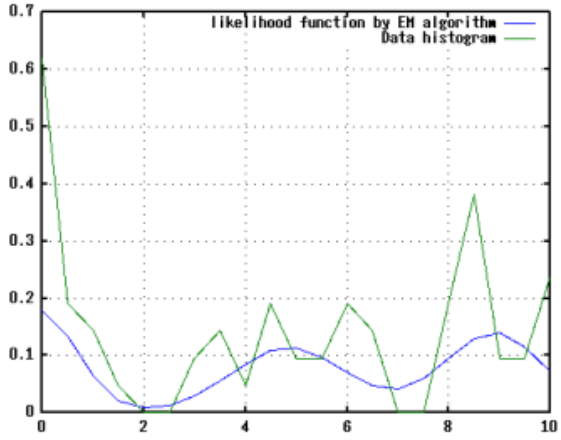

Рис. 2 Сумма гауссиан построенных по выборке

Примеры с кластеризацией данных с помощью EM алгоритма.

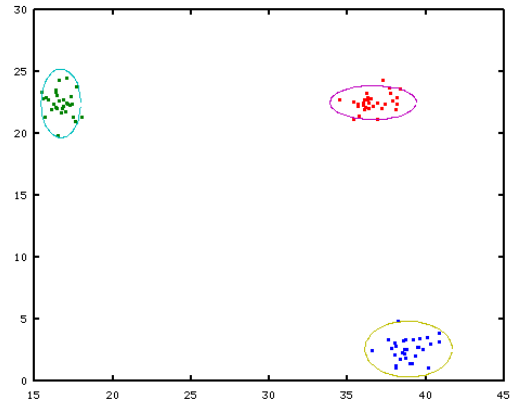

Рис. 3 Кластеризация

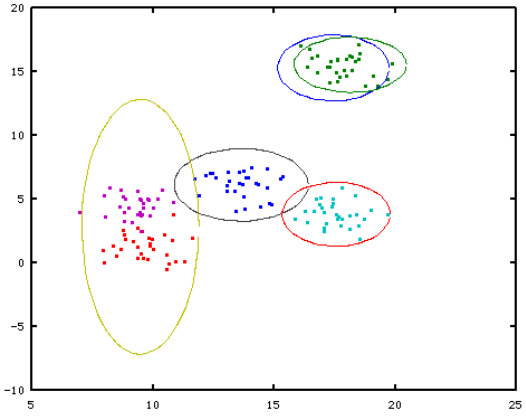

Рис. 4 Кластеризация

2.4.3. Моделирование фона.

Рассмотрим теперь моделирование фона видеоизображения: пусть $V_{n\times s}$ – область видеоизображения размером n на s. Распишем пошагово наши действия:

a) По нескольким первым кадрам с некоторым прореживанием собрать историю пикселей (набрать выборку) по области V.

Рис. 5 Схема сбора данных

b) Вычислить модель фона с помощью EM – алгоритма для выборки.

c) Для каждого пикселя области V последующих кадров:

c.1) Вычислить по некоторому порогу, принадлежит ли данный пиксель модели фона, если нет, то отнести этот пиксель к движению. Порогом для ответа о принадлежности фона может служить отклонение значения функции правдоподобия от минимального правдоподобия вычисленного для модели в случае, когда учитывается корреляция между RGB компонентами пикселя и ковариационная матрица не диагональна, либо правило "**s** сигм", где число s задаётся пользователем. В моделировании фона рекомендуется брать s равным 2.5.[1], [6].

c.2) Обновить модель фона по данному пикселю с помощью IIR фильтра [1] (фильтр с бесконечной импульсной характеристикой, рекурсивный фильтр).

c.3) Добавить данный пиксель в историю (выборку).

c.4) При необходимости пересчитать модель фона по области V.

2.4.4. Выбор начальных средних с помощью алгоритма k-means (k средних).

Проблемой EM алгоритма является, то что он может на некоторых наборах данных долго сходиться и давать плохие результаты. Поэтому важным является выбор начальных значений для вектора средних.

Рассмотрим применение алгоритма k-means к начальному выбору средних.[2], [5]. Действие алгоритма таково, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_j} (x_j - \mu_j)^2$$

где k – число кластеров, S_j – полученные кластеры, и μ_j – центры масс векторов

Алгоритм нахождения центров масс представляет собой версию EM-алгоритма, применяемого также для разделения смеси гауссиан. Он разбивает множество элементов векторного пространства на заранее известное число кластеров k. Основная идея заключается в том, что на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Алгоритм завершается, когда на какой-то итерации не происходит изменения кластеров. Это происходит за конечное число итераций, так как количество возможных разбиений конечного множество конечно, а на каждом шаге суммарное квадратичное отклонение V уменьшается, поэтому зацикливание невозможно.

Данный подход при подборе вектора средних даёт более быструю сходимость, но не защищает от попадания в локальный минимум при нахождении оптимальных параметров модели.

2.4.5. Оптимальный выбор средних для случая k = 2.

Для более быстрой и точной сходимости EM алгоритма для случая ограниченного пространства RGB можно выбрать начальное приближение для среднего более точно. Для этого рассмотрим одномерное пространство Y, где

$$Y = round(0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B), Y \in [0:255]$$

Будем считать, что Y принимает целые значения.

Если мы рассмотрим гистограмму нашей выборки пикселей в Y, нам интуитивно будет понятно где брать значения средних. Но как объяснить это компьютеру? Для начала найдём точку на гистограмме, где будет разделение гауссовых распределений. Гистограмму, которая приведена ниже, можно было бы разделить где-то на промежутке от 6 до 15, а потом выбрать за средние максимумы в точках 5 и 17, а потом обратно перейти в формат RGB, запомнив заранее какой нибудь пиксель попавший в выбранные начения Y. Для выбора точки разделения рассмотрим функцию

$$T(y) = F(x) \cdot y + (1 - F(y+1)) \cdot (255 - y),\ y \in [0:255]$$

где F(y) - функция распределения дискретной величины y.

Будем брать точкой разделения точку минимума функции $T(y)$. Точка минимума данной функции находится в области начала роста F(x) после промежутка с долгим убыванием или большого промежутка с одинаковыми значениями. Далее будем брать за первое среднее точку Y с наибольшим значением гистограммы слева от этой точки, а за второе - с наибольшим значением гистограммы справа от этой точки. Данный алгоритм можно обобщить на случай k больших 2-х, либо использовать данный алгоритм для пространства RGB.

Данный выбор средних даёт ускорение сходимости в несколько раз большее, чем при случайном выборе и выборе с помощью алгоритма k-means и во многих случаях позволяет точнее вычислять модель фона.

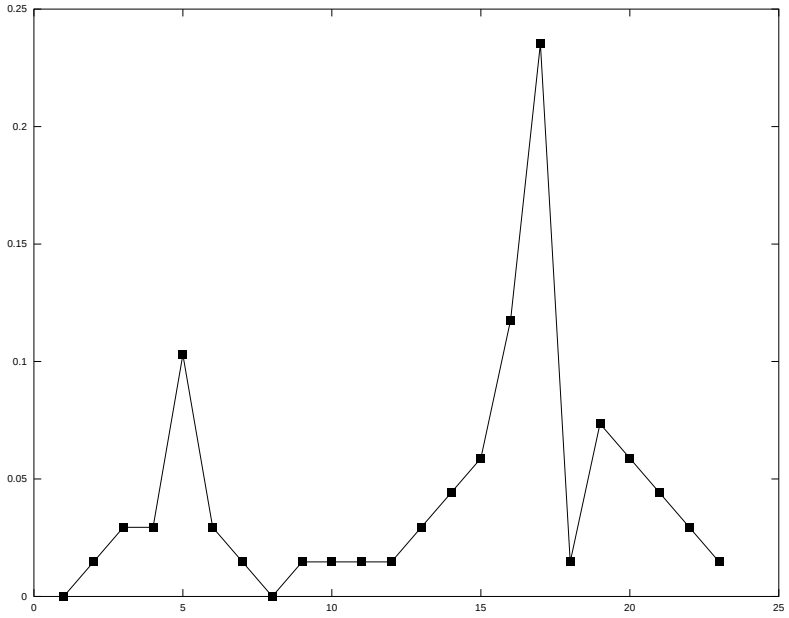

Рис. 6 Гистограмма данных

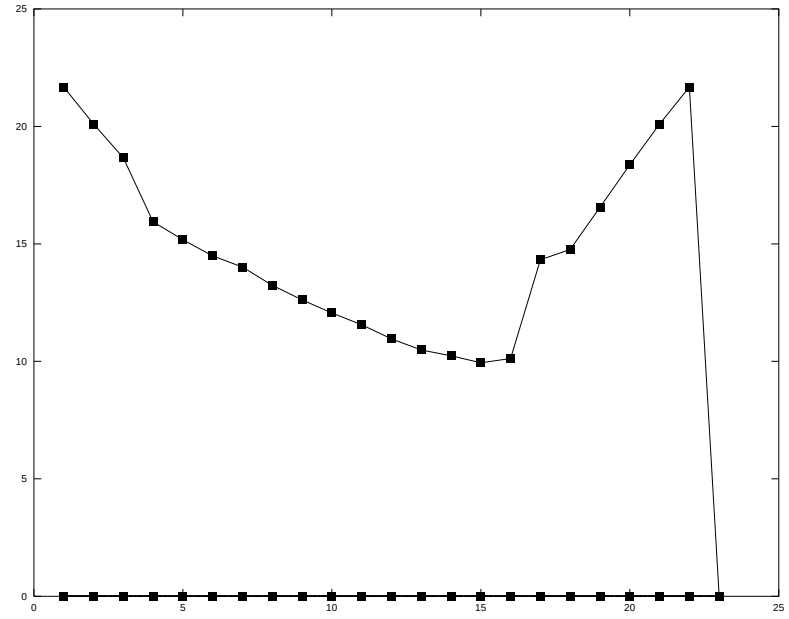

Рис. 7 Функция $T(y)$

Заметим, что данный алгоритм можно улучшать и модифицировать для достижения большей точности и высокой скорости работы.

2.5. Упрощение EM алгоритма для модели фона.

Можно заметить, что при учитывании корреляции между RGB компонентами пикселя, матрица ковариаций не всегда имеет диагональный вид, поэтому при вычислении принадлежности пикселя к модели применение правила трёх либо двух с половиной сигм ведёт к дополнительным вычислениям, поэтому можно упростить задачу считая ковариационную матрицу диагональной:

$$\Sigma = \sigma^2 I.$$

Данный подход позволяет значительно ускорить обработку кадров, так как при вычислении функции правдоподобия можно не использовать разложение холецкого и нахождение обратной матрицы и её определителя не составляет труда. При проверке на данных алгоритм в общем виде даёт результаты лучше, чем упрощённый алгоритм, но по скорости работы значительно уступает последнему.

2.6. Учитывание изменения фона по времени.

Понятно, что в видеоизображених снятых в долгий промежуток времени фон скорее всего будет сильно меняться из-за различной освещённости по времени. Для учитывания данного фактора можно по времени менять и саму модель фона введя ещё одну "лишнюю" гауссиану, в которой будут учитываться пиксели не попавшие в модель фона. Обновление данной гауссианы лучше всего проводить по следующим формулам:

$$\omega_k = b \cdot \omega_{k-1} + a \cdot M$$

$$\mu_k = b \cdot \mu_{k-1} + a \cdot x$$

$$\Sigma_k = b \cdot \Sigma_{k-1} + a \cdot (x - \mu_{k-1})(x - \mu_{k-1})^T$$

где **x** – вектор из RGB компонент, **b** и **a** обучающие коэффициенты модели, число **M** равно 1 если пиксель не попал в модель, иначе равно 0.

Если вес ω данной дополнительной компоненты смеси станет со временем больше, чем веса остальных компонент, то нужно ввести эту компоненту как основную, а компоненту с самым низким весом назначить как дополнительную.

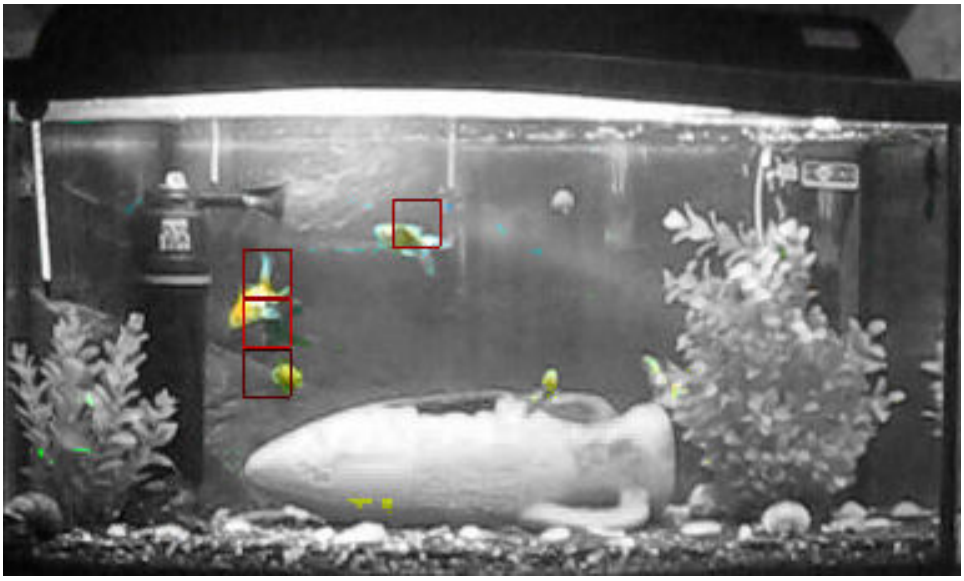

Рис. 8 Разделение на фон и "не фон".

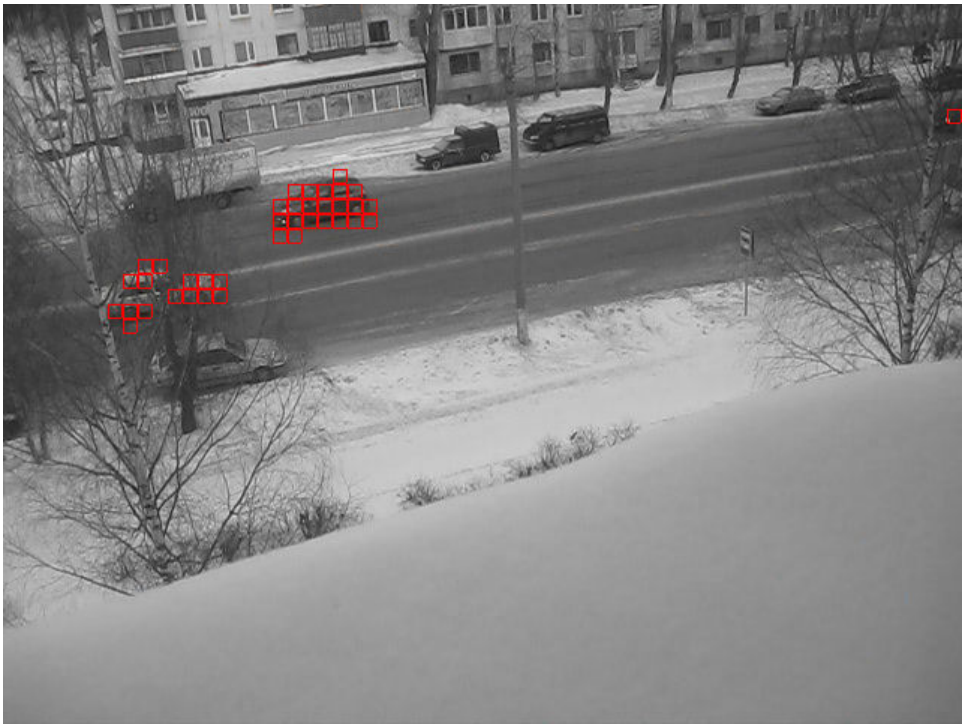

Рис. 9 Разделение на фон и "не фон".

3. Список литературы:

1) Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey. T. Bouwmans, F. El Baf, B. Vachon
2) The Expectation Maximization Algorithm. A short tutorial. Sean Borman
3) http://ru.wikipedia.org/wiki/EM-алгоритм
4) http://www.machinelearning.ru/wiki/index.php?title=EM_алгоритм_(пример)
5) http://ru.wikipedia.org/wiki/K-means
6) Adaptive background mixture models for real-time tracking. Chris Stauffer, W.E.L Grimson

Приложение

В данном приложении приведён пример вычисления смеси гауссиан с помощью EM алгоритма на языке системы для математических вычислений Octave.
Данная функция вычисляет компоненты смеси гауссиан для вектора данных x, k - число компонент, epsilon - порог для остановки итераций EM алгоритма.

```
function [omega, means, sigma, min_probability] = slow_em_algorithm(x, k, epsilon)
    s = size(x);
    len = s(2);

    dim  = s(1);

    probability = zeros(k, len);

    err = 20;
    omega = 1/k * ones(1, k);
    means=zeros(dim,k);
    sigma=zeros(dim,dim,k);

    for i=1:k
        sigma(:,:,i) = eye(dim,dim);
        means(:,i) = x(:,round(abs(rand()*(len-1))));
    endfor
    //expectation
    for i=1:len
        denum = 0;
        for j=1:k
            tmp = omega(j) * normal_distribution(x(:,i), means(:,j), sigma(:,:,j));
            probability(j,i) = tmp;
            denum += tmp;
        endfor
        for j=1:k
            probability(j,i) /= denum;
        endfor

    endfor
    loglike_prev = prob_expectation(omega, probability,k);
    while (1)
        //maximization
        for j=1:k
```

```
            omega(j) = new_omega(j, len,  probability);
            means(:,j) = new_mean(j, len, probability, dim, x, omega(j));
            sigma(:,:,j) = new_sigma(j, len, probability, dim, x, omega(j), means(:,j));
        endfor

        //expectation
        for i=1:len
            denum = 0;
            for j=1:k
                tmp = omega(j) * normal_distribution(x(:,i), means(:,j), sigma(:,:,j));
                probability(j,i) = tmp;
                denum += tmp;
            endfor

            for j=1:k
                probability(j,i) /= denum;
            endfor
        endfor

        loglike = prob_expectation(omega, probability,k)
        err = abs(loglike - loglike_prev)

        if (err < epsilon)
            break;
        endif
        loglike_prev = loglike;
    endwhile

    min_probability = Inf;

    for i = 1:len
        min_tmp = probability_i(omega, probability, k, i);
        if (min_probability > min_tmp)
            min_probability = min_tmp;
        endif
    endfor
end
function res = new_omega(j, len, probability)
    omega = 0;
    for i=1:len
```

```
            omega += probability(j, i);
        endfor

        if (omega > 0)
            res = omega / len;
        else
            res = omega;
        endif
end
function res = new_mean(j, len, probability, dim, x, omega_j)
        m = zeros(dim, 1);
        denumerator = omega_j * len;
        if (denumerator > 0)
            for i=1:len
                m += probability(j, i) * x(:,i);
            endfor
            m = m / denumerator;
        endif
        res = m;
end

function res = new_sigma(j, len, probability, dim, x, omega_j, mean_j)
        s = zeros(dim, dim);
        denumerator = 0.0;

        for i=1:len
            s += probability(j, i) * (x(:,i) - mean_j) * transpose(x(:,i) - mean_j);
            denumerator += probability(j,i);
        endfor
        s = s / denumerator;
        res = s;
end

function res = prob_expectation(omega, probability,k)
        res = 0;
        s=size(probability);
        len=s(1);
        for i=1:len
            tmp=0;
            for j = 1:k
```

```
                tmp += omega(j) * probability(j, i);
            endfor
            res += log(tmp);
        endfor
end

function res = probability_i(omega, probability, k, i)
        p = 0;
        for j=1:k
            p += probability(j, i) * omega(j);
        endfor
        res = p;
end

function res = normal_distribution(x_i, mean_i, sigma_i)
        div = sqrt(det(sigma_i));
        s=size(x_i);
        dim = s(1);
        a = 1 / ((2 * pi)^(dim / 2) * div);
        s = sigma_i^(-1);
        b = exp(-0.5 * transpose(x_i-mean_i)*s*(x_i-mean_i));
        res = a * b;
end
```