**Описание алгоритма работы.**

Ниже будет описан принцип работы алгоритма при анализе по одному параметру (например, цвету). На практике, нужно применять его по отдельности к каждому параметру, а потом накладывать полученные результаты друг на друга.

Будут использоваться следующие понятия:

* Исходное изображение имеет размер m x n пикселей
* Карта параметра - двухмерный массив m x n, в i,j-ом элементе которого хранится значение параметра ( **M** )
* Карта активности - двухмерный массив m x n, в i,j-ом элементе которого хранится число, пропорциональное тому, насколько "интересна" окрестность i,j-ого пикселя ( **A** )

**Алгоритм GBVS.**

Итак, задача в том, чтобы по карте параметра получить карту активности.

**Шаг 1. Построение цепи Маркова.**

После получения на вход карты параметров алгоритм полный ориентированный граф без петель так, что вес ребра (i, j)->(p, q) равен d((i, j)|(p, q))\*F(i - p, j - q), где d - логарифмическое расстояние между значениями параметра в (i, j) и (p, q), т.е.

d((i, j)|(p, q))= |log( **M**(i, j) / **M**(p, q))|,

F(a, b) = exp(-(a^2 + b^2) / (2o^2)), где o (сигма) - это некоторая константа.

После построения такого графа происходит его нормализация, т.е. для каждой вершины выбирается такой коэффициент, чтобы при домножении на него всех весов ребер, выходящих из вершины, сумма их была равна 1.

**Шаг 2. Получение равновесного распределения цепи Маркова.**

**Определение:** Имеется цепь Маркова, состоящая из n вершин (состояния системы) и ребер между ними (возможные переходы). Веса ребер соответствуют вероятности перехода по данному ребру при условии того, что система оказалась в вершине, из которой идет данное ребро. Каждому состоянию присваивается коэффициент Ki, равный вероятности оказаться в этой вершине при условии того, что от начала прошло много времени. Набор таких коэффициентов называется равновесным распределением.

Исходя из того, что Ki равен сумме w(a->i)\*Ka для всех a, отличных от i, получаем матрицу (mn)x(mn), решив которую, мы получим равномерное распределение, которое существует и единственно, с точностью домножения на число, для каждой системы\*. Если вспомнить, что вершины соответствуют пикселям картинки, то полученное распределение естественным образом представляется в виде двумерного массива. Получаем карту активности.

\*Из-за погрешностей арифметики чисел с плавающей точкой, иногда возникают ситуации, когда ядро матрицы составляет только нулевой вектор или, наоборот, его размерность больше 1. В первом случае я не знаю, что делать, кроме как уйти и плакать, а во втором случае нас устроит любой вектор-распределение из ядра, потому что они дают карты, достаточно близкие к искомой.

**Шаг 3. Нормализация карты активности.**

Определим одну итерацию нормализации. Сначала формируем цепь Маркова с весом ребра (i, j)->(p, q), равным **A**(p, q) \* F(i - p, j - q) без учета нормализации. Затем найдем равновесное распределение и запишем естественным образом в виде матрицы.

Итераций может быть несколько. Авторы используют 2-4 итерации.

**Идеи по улучшению.**

**Формирование неполного графа.**

В связи с тем, что веса ребер между далекими друг от друга пикселями намеренно делаются очень маленькими (за счет домножения на F), возникает вопрос, а нужны ли они вообще. Если при геометрическом расстоянии между позициями пикселей, превосходящем некоторое значение r (зависящее только от размеров исходного изображения), вместо того, чтобы добавлять очень легкое ребро, не добавлять вообще ничего, качество почти не страдает, а мы очень много выигрываем. А именно:

* Мы выигрываем время на вычислениях весов ребер, которые мы раньше делали бы, а теперь не делаем.
* Мы выигрываем очень много памяти, а которую раньше мы бы положили ребра, а теперь не будем.
* Мы выигрываем время на последующих переборах ребер.