

Лабораторная работа 6. СЕГМЕНТАЦИЯ ДВИЖУЩИХСЯ ОБЪЕКТОВ НА ОСНОВЕ ОЦЕНКИ ПОЛЯ ВЕКТОРОВ ДВИЖЕНИЯ

Для сегментации и сопровождения объектов по признаку движения может быть использована оценка поля векторов движения. Это поле описывает видимое перемещение объектов в кадре. Векторы движения позволяют не только детектировать факт перемещения объекта, но и оценить его скорость и направление.

Наличие информации о направлении и скорости позволяет решить большое число прикладных задач:

- сегментировать объекты на сложном фоне, находящиеся в непосредственной близости друг к другу;
- разрешить ситуации окклюзии при сопровождении путем выявления объекта, находящегося на переднем плане;
- построить модель движения по совокупности векторов (на основе информации, полученной в одном кадре) и осуществить сопровождение объекта.

Аппарат векторов движения (векторов оптического потока) был разработан для применения в стандартах видеокompрессии *MPEG*. Однако, в видеоаналитике векторы движения также стали очень популярны и востребованы. В задачах компьютерного зрения вектор движения показывает смещение фрагмента изображения в кадре $t+1$ относительно кадра t .

Для определения векторов движения основным является уравнение оптического потока, полученное на основе допущения о постоянстве яркости $L(x,y,t)$ точки (пикселя) при движении

$$\frac{d}{dt}L(x,y,t)=0.$$

Так как $x=f(t)$ и $y=f(t)$, вычисления нужно проводить по формуле сложной производной:

$$\frac{d}{dt}L(x,y;t)=\frac{\partial L}{\partial x} \cdot \frac{\partial x}{\partial t} + \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial t} + \frac{\partial L}{\partial t} = \langle \nabla L, \mathbf{v} \rangle + \frac{\partial L}{\partial t} = 0,$$

где $L(x,y,t)$ – яркость пикселя с координатами x и y в момент времени t ;

$\langle \dots \rangle$ обозначает скалярное произведение векторов, а $\nabla \mathbf{L} = \left(\frac{dL}{dx}, \frac{dL}{dy} \right)^T$ – вектор-градиент; $\mathbf{v} = [v_x, v_y]^T$ – вектор скорости (оптического потока).

Вместо производных по времени и пространству используют их целочисленные приближения

$$dL/dx \approx \Delta L / \Delta x; \quad dL/dy \approx \Delta L / \Delta y; \quad dL/dt \approx \Delta L / \Delta t$$

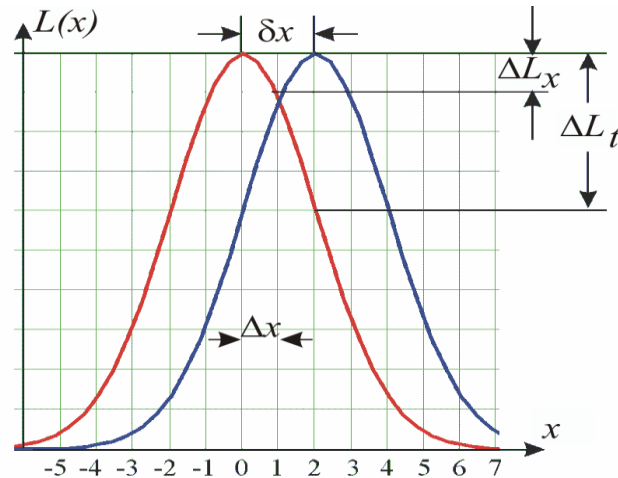


Рис. 6.1. Иллюстрация к расчету вектора движения

и вычисляют приращения яркости в соседних пикселях по вертикали и горизонтали, а также в соседнем кадре, т. е. приравнивают Δx и Δy одному пикселю, а Δt – одному кадру (рис. 6.1).

Из анализа уравнения оптического потока можно сделать следующие выводы (которые объясняют ограничения метода):

1. Уравнение оптического потока является недоопределенным и позволяет найти только сонаправленную с яркостным вектором-градиентом

$\nabla \mathbf{L} = \left(\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y} \right)$ компоненту векторов оптического потока. Ортогональная с

яркостным вектором-градиентом компонента может принимать любые значения, не изменяя скалярного произведения, и поэтому не может быть определена однозначно.

Для полной оценки векторов оптического потока необходимо ввести требование гладкости – близости скоростей у группы соседних пикселей. Принятие решения о размере такой группы называют проблемой апертуры.

2. Однозначное определение векторов оптического потока возможно только в случае, если компоненты яркостного вектора градиента отличны от нуля, т.е. имеют место изменения яркости по горизонтали и вертикали. В случае гладкой поверхности достоверную оценку векторов найти нельзя.

3. Уравнение оптического потока предполагает постоянство яркости при движении точки вдоль траектории. Подсветки, тени, блики (которые часто возникают при видеонаблюдении на открытом воздухе), низкая детальность, прозрачные и зеркальные поверхности реальных объектов нарушают это утверждение, что затрудняет вычисления и приводит к ошибкам при определении векторов оптического потока.

Перечисленные факторы обуславливают высокую вероятность появления ошибочных, так называемых «аномальных векторов», которые не соответствуют реальному перемещению, существующему в видеопоследовательности, и могут быть не сонаправленными с движением объекта, к которому они принадлежат. Аномальные векторы в задачах видеокомпрессии приводят только к уменьшению степени сжатия, в то время, как в системах видеоаналитики их следствием является разделение объектов интереса на части (ошибка сегментации), снижение точности моделей движения, потеря объекта при сопровождении.

Для поиска векторов движения существует три группы методов: дифференциальные, фазовые и корреляционные.

Дифференциальные методы определяют векторы оптического потока исходя из предположения, что изображение является непрерывным (дифференцируемым) в пространстве и во времени. Методы этой группы делятся на глобальные и локальные, а также на методы первого и второго порядка на основании используемых производных.

Глобальные методы используют основное уравнение оптического потока и добавляют к нему некую функцию ошибок. В известном методе Хорна-Шунка, оптический поток определяют путем минимизации функционала

$$\int_S \left(\langle \nabla \mathbf{L}, \mathbf{V} \rangle + L_t \right)^2 + \lambda^2 \text{tr} \left((\nabla \mathbf{V})^T, (\nabla \mathbf{V}) \right) d\mathbf{r} = \min,$$

$$\nabla \mathbf{V} = \left(\frac{d^2 x}{dt^2}, \frac{d^2 y}{dt^2} \right),$$

где первое слагаемое – уравнение оптического потока, а второе – функция ошибок, учитывающая близость скоростей; λ – заранее заданная константа; $\text{tr}(\dots)$ – след матрицы, равный сумме ее диагональных элементов; S – область изображения, для которой ищут минимум функции; $\mathbf{r} = (x, y)^T$ – вектор пространственных координат; L_t – обозначена производная яркости по времени.

Преимуществом метода Хорна-Шунка является высокая плотность получаемых векторов движения. Для оценки векторов, принадлежащих внутренним гладким частям объекта, используются значения, полученные для хорошо выделяющихся движущихся внешних границ областей объекта. К недостаткам относят относительно низкую устойчивость к шуму (по сравнению с локальными методами).

Локальные методы основаны на предположении, что в окрестности каждого пикселя значение оптического потока одинаково, таким образом можно записать основное уравнение оптического потока для группы пикселей (маска S) окрестности и решить полученную систему уравнений методом наименьших квадратов.

Минимизируют квадратичную функцию ошибки

$$\sum_{x,y \in S} \sum (\langle \nabla L(x, y, t), \mathbf{v} \rangle + L_t(x, y, t))^2 = \min.$$

Для взвешенного метода наименьших квадратов

$$\sum_{x,y \in S} [\mathbf{W}(x, y)] (\langle \nabla L(x, y, t), \mathbf{v} \rangle + L_t(x, y, t))^2 = \min,$$

где $\mathbf{W}(x, y)$ – диагональная весовая матрица, усиливающая влияние центральных пикселей маски S при оценке векторов оптического потока.

Элементы на диагонали матрицы $w(x, y) = w(x) \otimes w(y)$, где знак \otimes означает кронекерово произведение (каждый с каждым). Например, при маске S (5x5) пикселей, принимают $w(-2) = w(2) = 0,0625$; $w(-1) = w(1) = 0,25$; $w(0) = 0,375$, т. е. каждому из 25 пикселей изображения в окне присвоен весовой коэффициент.

Оценка векторов движения определяется соотношением:

$$\mathbf{v} = [\mathbf{A}^T \mathbf{W} \mathbf{A}]^{-1} \mathbf{A}^T \mathbf{W} \mathbf{b},$$

Где

$$\mathbf{A} = \begin{bmatrix} L_x(x-2, y-2, t) & L_y(x-2, y-2, t) \\ L_x(x-2, y-1, t) & L_y(x-2, y-1, t) \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ L_x(x+2, y+1, t) & L_y(x+2, y+1, t) \\ L_x(x+2, y+2, t) & L_y(x+2, y+2, t) \end{bmatrix};$$

$$\mathbf{W} = \begin{bmatrix} w(-2, -2) & & & & & & \\ & w(-2, -1) & & & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & \cdot & & \\ & & & & & w(2, 1) & \\ & & & & & & w(2, 2) \end{bmatrix};$$

$$\mathbf{b} = \begin{bmatrix} L_t(x-2, y-2, t) \\ L_t(x-2, y-1, t) \\ \\ \\ L_t(x+2, y+1, t) \\ L_t(x+2, y+2, t) \end{bmatrix}.$$

Локальный дифференциальный метод Лукаса-Канаде, базирующейся на вышеописанных принципах, практически является самым популярным алгоритмом для вычисления оптического потока в системах компьютерного зрения. Алгоритм Лукаса-Канаде менее чувствителен к шуму на изображениях, чем глобальные методы, однако он не может определить направление движения пикселей внутри однородных областей.

Фазовые алгоритмы основаны на методе фазовой корреляции для сопоставления изображений (в данном случае – фрагментов изображений, блоков). Метод использует переход в частотную область для вычисления взаимной спектральной плотности и обратное преобразование для получения функции кросс-корреляции, максимум которой позволяет определить смещение, то есть вектор движения.

Корреляционные методы определяют векторы оптического потока на основе смещений, при которых достигается максимальное соответствие фрагментов изображения текущего и предыдущего кадров. Определение наилучшего соответствия выполняется путем поиска максимума корреляционной функции.

Наиболее часто используют метод сопоставления блоков, принятый во многих стандартах видеокодирования.

Метод состоит из следующих основных шагов:

- текущий кадр делится на неперекрывающиеся квадратные блоки размером $M \times N$ пикселей;
- для каждого блока формируется область поиска в предыдущем кадре, которая имеет размер $(2d+M+1) \times (2d+N+1)$ пикселей, где d – это максимально

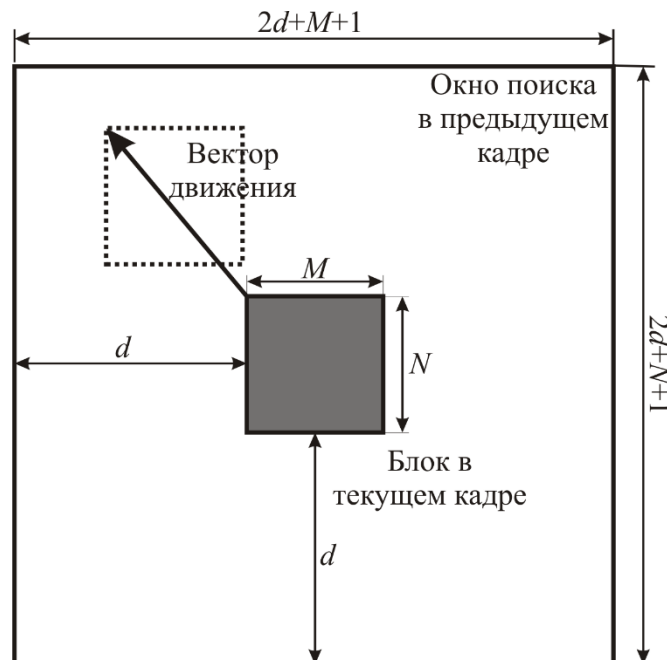


Рис. 6.2. Совмещение блоков

возможное смещение в горизонтальном и вертикальном направлениях (рис. 6.2);

- выполняется совмещение блоков текущего кадра с блоками предыдущего кадра видеопоследовательности, определяется, какому блоку в области поиска текущий блок лучше всего соответствует, и оценивается величина смещения положения блока в текущем кадре относительно предыдущего – вектор движения.

Считается, что все пиксели блока претерпевают одинаковое перемещение и им приписывается один и тот же вектор движения.

Задача определения векторов движения в этом случае решается путем минимизации целевой функции, характеризующей степень соответствия (совпадения) двух блоков, на множестве различных положений обрабатываемого блока в области поиска.

Формирование целевой функции, оценивающей степень соответствия между блоком текущего кадра и блоком предыдущего кадра, может быть выполнено в нескольких вариантах:

1) средняя абсолютная разность (*MAD*):

$$MAD(v_x, v_y) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |L_c(x_k + i, y_l + j) - L_p(x_k + v_x + i, y_l + v_y + j)|,$$

где $L_c(\dots)$ и $L_p(\dots)$ – яркости пикселя в текущем и предыдущем кадре соответственно; (x_k, y_l) – координаты пикселя левого верхнего угла текущего блока; $N \cdot N$ – размер блока; (v_x, v_y) – один из возможных векторов движения;

2) среднеквадратическая ошибка (*MSE*):

$$MSE(v_x, v_y) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (L_c(x_k + i, y_l + j) - L_p(x_k + v_x + i, y_l + v_y + j))^2;$$

3) нормированная функция взаимной корреляции (*NCCF*):

$$NCCF(v_x, v_y) = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} L_c(x_k + i, y_l + j) * L_p(x_k + v_x + i, y_l + v_y + j)}{\sqrt{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} L_c^2(x_k + i, y_l + j) * \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} L_p^2(x_k + v_x + i, y_l + v_y + j)}};$$

4) максимальное число соответствующих пикселей (*MPC*):

$$MPC(v_x, v_y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} T(x_k + i, y_l + j, v_x, v_y),$$

$$T(s, t, v_x, v_y) = 1, \text{ если } \lfloor L_c(s, t) - L_p(s + v_x, t + v_y) \rfloor < Th, \text{ иначе } T(s, t, v_x, v_y) = 0,$$

где Th – предопределенный порог.

В отличие от предыдущих критериев блок наилучшего соответствия, найденный с использованием *MPC*, тот, который дает самое большое значение целевой функции. Обычно в качестве целевой функции используется *MAD*, так как она дает характеристику, близкую к характеристике *MSE*, но не требует операций умножения.

Совмещение блоков. Самым простым и надежным алгоритмом, позволяющим выполнить совмещение блоков, является полный перебор *FS* (*full search*), но из-за большого объема вычислений он обладает низкой скоростью.

Разработано большое число алгоритмов, которые направлены на оптимизацию стратегии поиска минимума. Их можно разделить на два класса:

- алгоритмы, уменьшающие число вычислений при определении целевой функции;
- алгоритмы, уменьшающие число контрольных точек в области поиска.

Примерами, относящимися к первому классу, являются алгоритм неполного определения целевой функции, алгоритм остановки на полпути (*PDS*) и остановки на полпути с нормировкой (*NPDS*).

В рамках второго класса можно выделить три группы алгоритмов:

1. основанные на свойстве унимодальности целевой функции; примерами являются алгоритм поиска тремя итерациями (*3SS*), алгоритмы логарифмического, ортогонального и поперечного поиска, алгоритм поиска по квадрантам;
2. учитывающие одновременно унимодальность минимизируемой функции и скорость оцениваемого движения; к этой группе относятся алгоритмы двух видов: ориентированные на работу с видеопоследовательностями, в которых преобладает медленное движение, такие как блочный градиентный поиск, четырехшаговый

алгоритм (4SS), а также производящие предсказание характера оцениваемого движения (быстрое/медленное) и далее использующие наиболее эффективный для данного вида движения подход, – гибридные алгоритмы.

3. предсказывающие начальное приближение: алгоритмы с предсказанием и иерархический поиск.

В качестве примера рассмотрим работу алгоритма 3SS. Многие из вышеперечисленных подходов будут отличаться от него только расположением точек для вычисления значений целевой функции, или числом шагов.

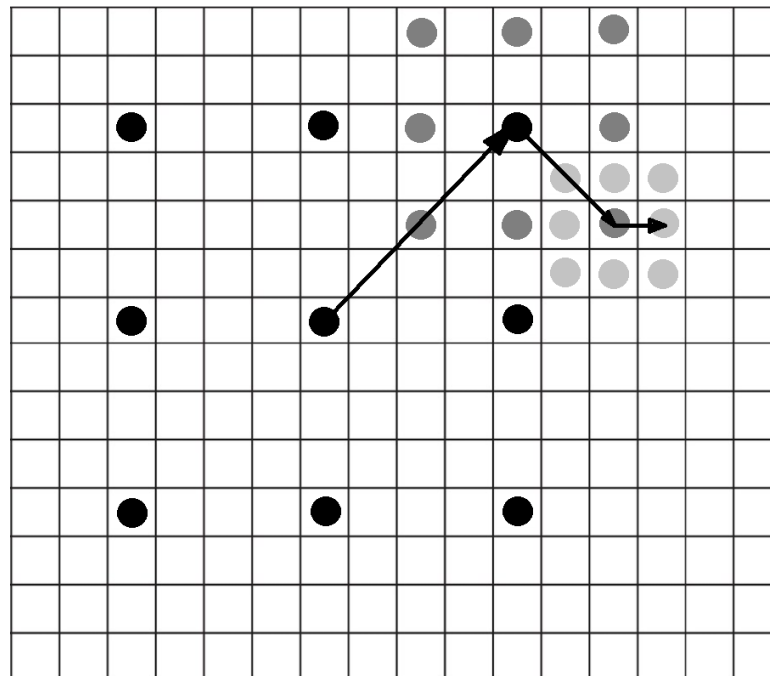


Рис. 6.3. Иллюстрация алгоритма 3SS.

Для каждого блока в кадре t на первом этапе вычисляется целевая функция (MAD , MSE , или $NCCF$) в девяти точках, то есть мера корреляции с девятью блоками в кадре $t+1$ (черные точки на рис. 6.3), отстоящих друг от друга на некоторый шаг d и формирующих квадрат. Затем, уменьшив шаг d вдвое, вычисляют еще 8 значений целевой функции относительно точки, принадлежащей блоку, который имеет наибольшую корреляцию с блоком кадра t – минимум целевой функции (темно-серые точки на рис. 6.3). На последнем, третьем, шаге операцию повторяют, снова уменьшив шаг d в два

раза. В результате, для каждого блока в кадре t находят наиболее похожий на него блок в кадре $t+1$ и соответственно определяют вектор движения.

Разумеется, ни один алгоритм не гарантирует безошибочного нахождения векторов движения. На практике негативное влияние на корректность определения векторов могут оказывать шумы, блок может сместиться на расстояние, выходящее за границы области поиска. Все эти факторы приводят к появлению ошибочно найденных векторов, которые затрудняют сегментацию.

Для снижения влияния аномальных векторов движения на результат обработки, полученное поле векторов целесообразно обработать – выполнить пространственную или временную фильтрацию.

Рекурсивная векторная медианная фильтрация обеспечивает высокую эффективность обработки поля векторов в пределах одного кадра (рис 6.4).

Под медианой множества векторов понимается такой вектор из рассматриваемого множества, у которого сумма расстояний до всех других минимальна. Расстояние между двумя векторами, $\mathbf{U}(x_u, y_u)$ и $\mathbf{V}(x_v, y_v)$ вычисляется на основе нормы L_2 :

$$\|\mathbf{U} - \mathbf{V}\|_{l_2} = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2}$$

В применении к задаче удаления аномальных векторов под медианной фильтрацией понимается замена каждого вектора движения векторной медианой множества, составленного из самого вектора и восьми его ближайших соседей. При вычислении расстояний используются только ненулевые векторы, т.е. каждый ненулевой вектор превращается в векторную медиану, вычисленную с помощью его восьми ненулевых соседей (рис. 6.4). Это делается для того, чтобы избежать замены ненулевых векторов на нулевые, когда в этом соседстве доминируют нулевые векторы.


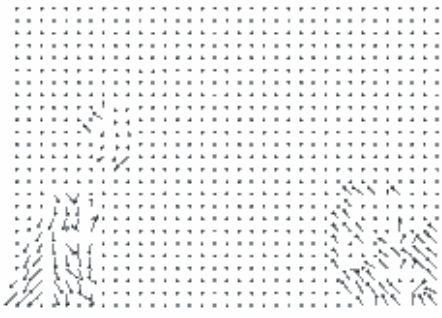
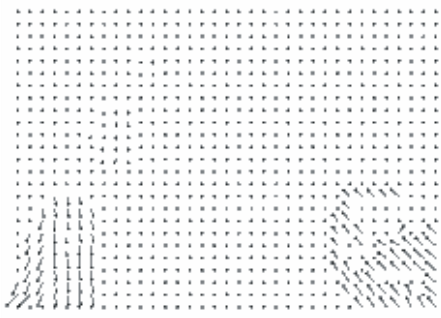

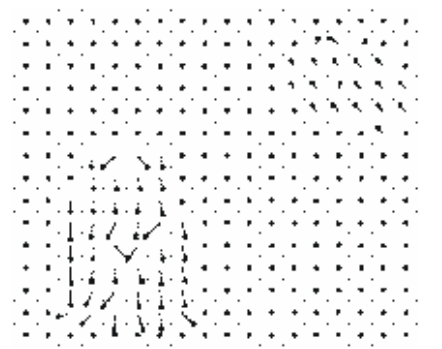
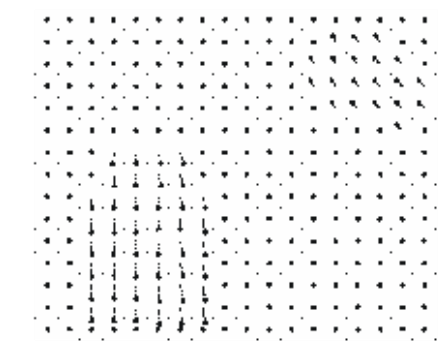

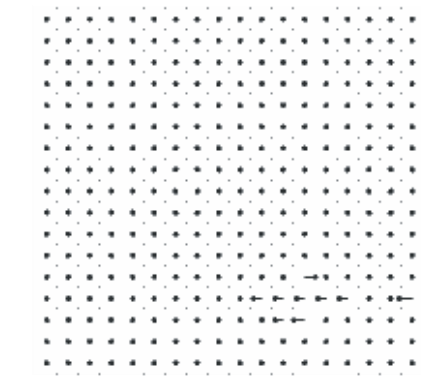
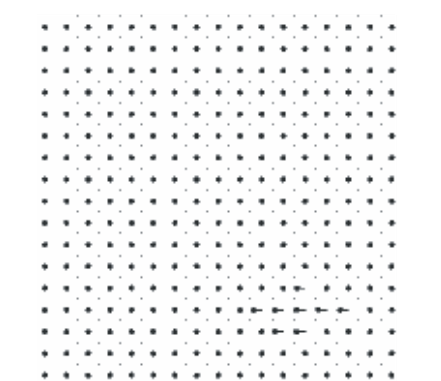
Вид кадра	Исходное поле векторов	Поле векторов после фильтрации
		
		
		

Рис. 6.4. Медианная фильтрация полей векторов движения

Пространственная медианная фильтрация поля векторов движения позволяет повысить точность сегментации объектов интереса за счет устранения аномальных векторов движения и вызванных ими разрывов внутри

сегментированного объекта. Однако объекты, которые по размерам меньше маски медианного фильтра, могут быть потеряны.

На заключительном этапе алгоритма сегментации объектов на основе векторов движения найденные векторы, прошедшие фильтрацию, объединяют в кластеры по признаку сонаправленности и пространственной связанности. **Алгоритм кластеризации** состоит из следующих шагов.

1. Последовательно перебирают все блоки изображения, для блоков с ненулевым вектором движения проверяют наличие метки, если метка отсутствует, то для данного блока формируют новую метку.
2. Вокруг блока с созданной меткой рассматривают 8 его соседей. Если встречается ненулевой вектор и принимается решение, что векторы сонаправленные, то текущая метка присваивается соседнему блоку и шаг 2 повторяется. В противном случае повторяют шаг 1.

Процедуру выполняют до тех пор, пока существуют неразмеченные блоки. Результатом обработки являются сегментированные движущиеся объекты.

Таким образом, алгоритм сегментации объектов на основе векторов движения состоит из следующих шагов:

- разбиение изображения на блоки;
- определение векторов движения;
- пространственная фильтрация поля векторов движения;
- кластеризация блоков, для которых определены ненулевые векторы движения, определение их принадлежности к соответствующим объектам.

Совмещение с шаблоном. Вспомогательные материалы для выполнения лабораторной работы

Совмещение с шаблоном — общее название техник обработки изображений, связанных с поиском определенного фрагмента на снимке, или видео. Во втором практическом задании это может быть полезно при поиске векторов движения полным перебором. Шаблоном в данном случае будет

являться блок из первого изображения, для которого мы ищем соответствующий блок на втором.

Для выделения (вырезания) блоков удобно воспользоваться структурой *cv::Rect* (*x*, *y*, *width*, *height*). Например, можно вырезать небольшой фрагмент из изображения и сохранить его для дальнейшей работы. Для этого надо при инициализации нового объекта класса *Mat* в качестве аргумента указать ограничивающий прямоугольник.

```
int main()
{
    Mat img1 = imread("C:/../tulips.jpg");
    cv::Rect rect(100, 100, 200, 200);

    Mat img2, img3;
    img2 = img1(rect);
    img3 = img1(rect).clone();

    imshow("img1", img1);
    imshow("img2", img2);
    imshow("img3", img3);
    waitKey();

    return 0;
}
```

Приведенный код иллюстрирует (рис. 6.5) пример вырезания фрагментов из исходного изображения. Для этого создается прямоугольник *rect* с координатами верхнего левого угла (*100,100*) и длиной и шириной, равным 200. Затем с его помощью инициализируются два новых изображения.

Важным отличием является то, что *img3* за счет использования функции *clone()*, является независимой «глубокой копией» соответствующего фрагмента *img1*. Изменяя его, программист не повредит исходный снимок.

Img2, напротив, ссылается на ту же область памяти, что и фрагмент *img1*. Внесенные изменения в *img2* отразятся и на оригинале.

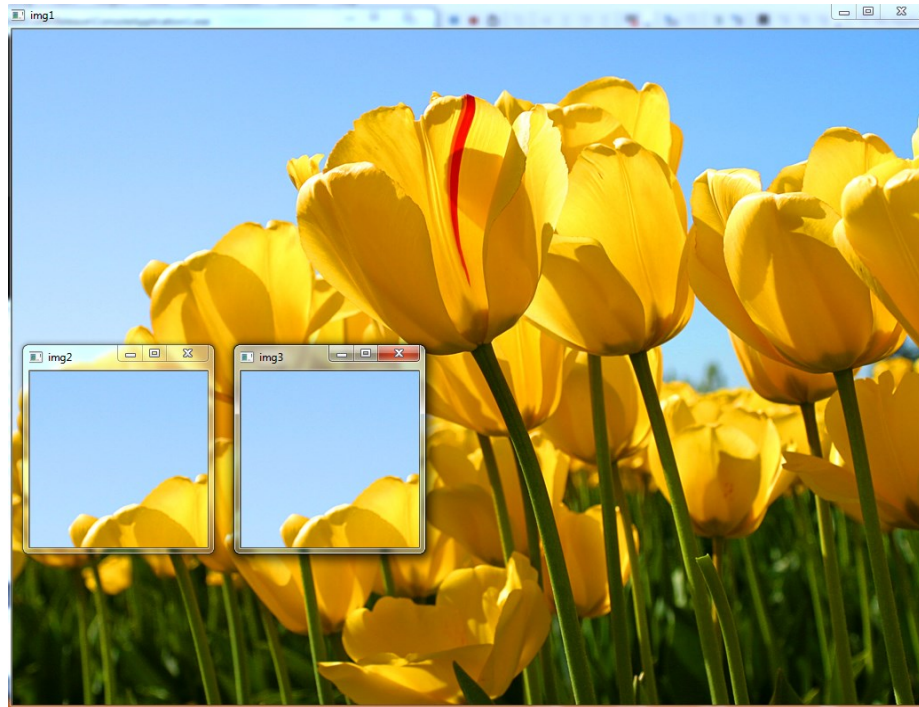


Рис. 6.5. Вырезание части изображения

Поиск шаблона на изображении рассмотрим на примере – найдем координаты вырезанного прямоугольника на исходной картинке. Для этого необходимо предварительно создать объект класса *Mat* для сохранения результата, выполнить функцию OpenCV *cv::matchTemplate* и определить координаты совпадения с помощью функции *cv::minMaxLoc*.

```
#include "stdafx.h"
#include <imgproc.hpp>
#include <highgui.hpp>
#include <iostream>

using namespace cv;

int main()
{
    Mat img1 = imread("C:/../tulips.jpg");
    cv::Rect rect(100, 100, 200, 200);

    Mat img2, result;

    img2 = img1(rect).clone();

    //Определяем размеры матрицы - результата
    int result_cols = img1.cols - img2.cols + 1;
```

```

int result_rows = img1.rows - img2.rows + 1;

result.create(result_rows, result_cols, CV_32FC1);

/// Сопоставление с шаблоном и нормировка результата
matchTemplate(img1, img2, result, CV_TM_SQDIFF);
normalize(result, result, 0, 1, NORM_MINMAX, -1, Mat());

/// Определение точки наилучшего соответствия
double minVal; double maxVal; Point minLoc; Point maxLoc;
minMaxLoc(result, &minVal, &maxVal, &minLoc, &maxLoc, Mat());

std::cout << minLoc << std::endl;

imshow("img1", img1);
imshow("img2", img2);
imshow("result", result);

waitKey();

return 0;
}

```

Результат приведен на рисунке 6.6. В данном случае матрица *result* инициализируется с помощью метода *create* – это еще один возможный вариант создания объекта. Тип *CV_32FC1* в явном виде указывает на то, что нам требуется одноканальная матрица с элементами типа *float*. Четвертый аргумент функции *cv::matchTemplate* – метод сопоставления, вернее, применяемая метрика. В данном случае выбрана среднеквадратическая ошибка, но может быть использован и коэффициент корреляции (*CV_TM_CCORR*), и другие метрики. Функция нормировки приводит все значения к диапазону от нуля до единицы – это в том числе позволяет вывести изображение формата *CV_32FC1* на экран.

Как видно на скриншоте консольного вывода, координаты были определены правильно – это точка *(100, 100)*. На изображении-результате этим координатам соответствует самый черный пиксель, что логично, так как в этой точке значение среднеквадратической ошибки минимально.

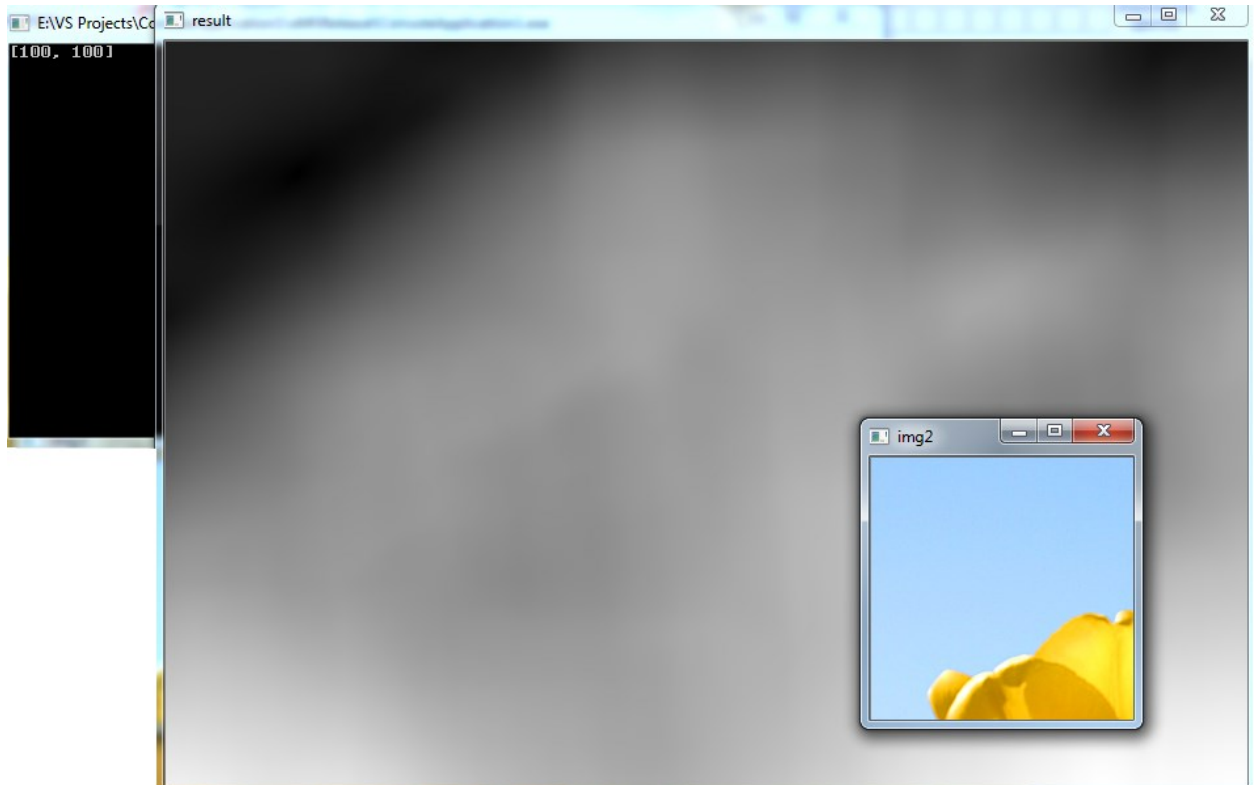


Рис. 6.6. Результат сопоставления с шаблоном.

Задание на моделирование

Разработать программу, моделирующую алгоритм сегментации объектов на основе векторов движения. Исходными данными являются два кадра видеопоследовательности со статичным фоном, на которых присутствует пара разнонаправленно перемещающихся объектов. В программе должны быть реализованы следующие процедуры с выводом результата на экран:

- загрузка пары кадров,
- разбиение кадров на блоки,
- реализация корреляционного алгоритма поиска векторов движения (с выводом векторов поверх изображения), с использованием в качестве целевой функции по указанию преподавателя MAD, или MSE (среднюю квадратичную ошибку) и одну из процедур для организации поиска (по указанию преподавателя): полный перебор, 3SS, 4SS, логарифмический, или ортогональный поиск.

- фильтрация поля векторов движения (демонстрация результата на изображении),
- кластеризация векторов и сегментация объектов (с отображением результата на кадре).

Контрольные вопросы

1. Дайте определение векторам движения.
2. На основании какого предположения сформировано основное уравнение оптического потока?
3. Перечислите основные ограничения уравнения оптического потока.
4. Какие группы методов нахождения векторов движения вы знаете?
5. Приведите пример корреляционного метода.