Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Лабораторная работа №1

Определение векторов смещения

Дисциплина: Разработка графических приложений

Выполнил студент гр. 13541/3 _____ Ерниязов Т.Е.
(подпись)
Руководитель _____ Абрамов Н.А.
(подпись)

Санкт – Петербург
2018

Цель работы:

Реализовать алгоритм поиска векторов смещения на изображения, используя метрику SAD.

Описание алгоритма поиска смещения на изображении

1. Выбор двух последовательных изображений
2. Разбиение изображения на блоки
3. Поиск похожих блоков первого изображения во втором с помощью метрики SAD методом полного перебора
4. Визуализация векторов смещения, наложенных на исходное изображение

Эксперименты:

Нахождение и отображение векторов смещения на изображениях:

- Два последовательных кадра из нескольких роликов
- Смещение изображение вправо

Работа алгоритма

Алгоритм использует схожесть соседних кадров в видео последовательности и находит векторы движения отдельных частей изображения.

Этапы:

- Загружаются два последующих кадра.
- Кадры разбиваются на блоки 8×8 пикселей
- Производится обход блоков (каждый блок в данном случае обрабатывается отдельно).
- При счете одного блока производится обход некоторой окрестности блока в поиске максимального соответствия изображению блока на предыдущем кадре в пределах этой окрестности.
- Таким образом, после завершения поиска мы получаем набор векторов, указывающий «движение» блоков изображения между кадрами. Эти векторы могут быть естественным образом использованы для создания изображения скомпенсированного кадра, который лучше приближает кадр, для которого производилась компенсация движения.

Ход работы

Рассмотрим эксперимент, в котором будем определять вектора смещений для 2 последовательных кадров.

Рис. 1. Два последовательных кадра.

В качестве результата получаем, изображение с отображением рассчитанных векторов смещения. Размер изображений 1280x800 пикселей.

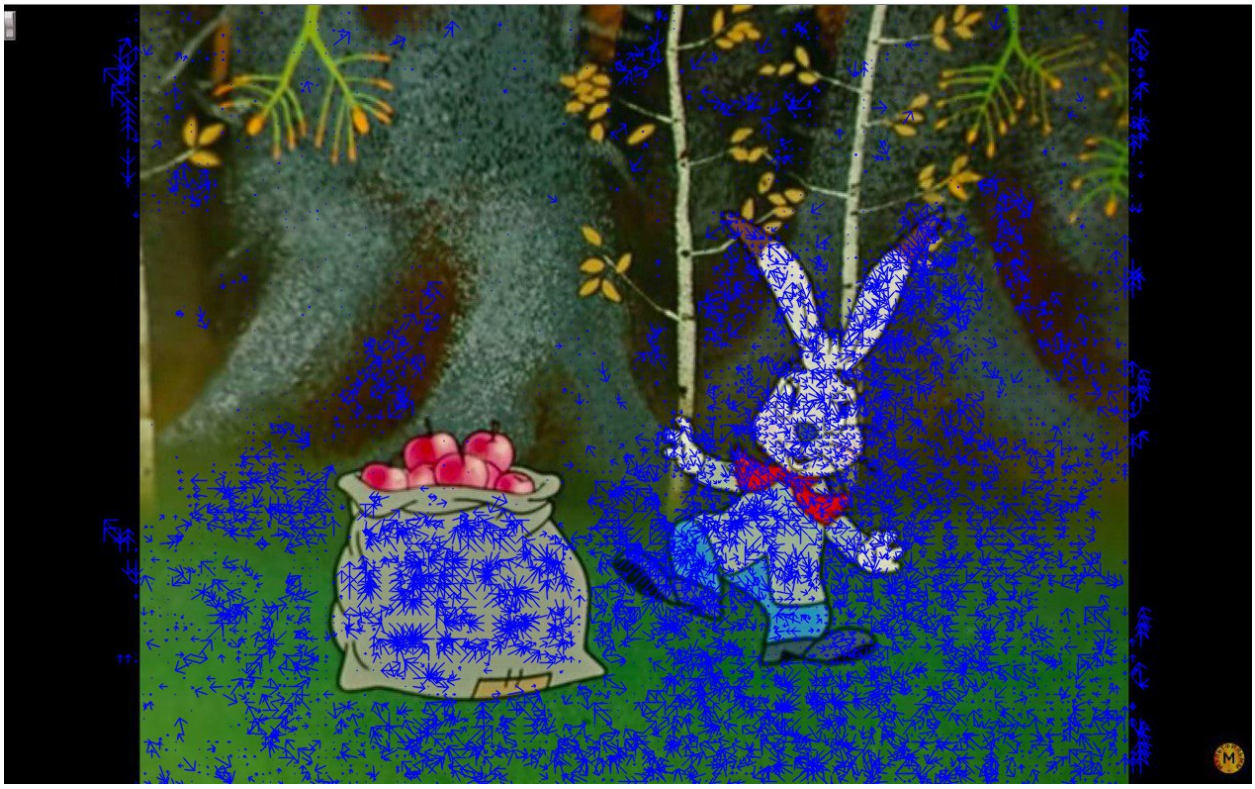

Рис. 2. Скомпенсированный кадр с векторами движения для каждого блока (точка — это нулевой вектор)

Из-за большого разрешения изображения, не получается разглядеть работу алгоритма. Векторов движения получается слишком много.

Попробуем сжать изображение. При небольшом сжатии (мы возьмем 36 процентов от оригинального изображения) признаки не должны пропасть, а вектора будут более обозначены.
Повторим эксперимент, с другими последовательными кадрами и возьмем 36 процентов от оригинального размера:

Рис. 3. Два последовательных кадра.

В результате получаем следующее изображение:

Рис. 4. Скомпенсированный кадр с векторами движения для каждого блока (точка — это нулевой вектор) на изображении с уменьшенной размерностью

Теперь легче наблюдать за работой алгоритма. Фон на видео не меняться (вектора смещения отсутствуют), а на том участке изображения, где происходит движение (руки и ракетка) мы наблюдаем верно направленные вектора смещения.

Попробуем провести еще один эксперимент. Возьмем два последующих кадра, из видео, где камера плавно двигается. В данном случае, следит за бегунами.

Рис. 5. Два последующих кадра

Снова для представления лучшего результата, возьмем 36 процентов от оригинального изображения. Получим следующий результат:

Рис. 6. Кадр с векторами движения для каждого блока на изображении со смещением сцены

В результате, видим верно определенные вектора смещения, значит алгоритм работает. Изображения были перепутаны местами (как при реверсивной съемке), но нам это не важно, нам важен сам факт смещения, поэтому вектора идут от правого верхнего угла к левому нижнему, хотя движение камеры было от левого нижнего угла к правому верхнему.

Вывод

С помощью данного алгоритма возможно качественно определить вектора смещения на изображении.

Однако на этапе выбора материалов для проведения экспериментов, было выявлено, что на однородных участках изображений однозначно определить вектор смещения не удается.

Также следует отметить, что и после выбора материалов экспериментов однородность изображения сказывалась на результате, в каждом из продемонстрированных можно увидеть недостатки связанные с однородностью изображения. Тем не менее не следует признавать данные недостатки критическими.

Неприемлемые результаты расчета векторов смещения объясняются тем, что возможно исчезновение частей изображения от кадра к кадру (под эту ситуацию подходят в частности границы), а также низкое устойчивость к шуму и изменение яркости изображения.

Код программы

```
void Motion_Compensation(cv::Mat mat1, cv::Mat mat2)
{

    // инициализируем размер изображений
    int height_1 = mat1.size().height;
    int width_1 = mat1.size().width;
    int height2 = mat2.size().height;
    int width2 = mat2.size().width;

    CvPoint a;
    CvPoint b;

    // инициализируем размер блоков разбиения и окрестности SAD
    int BLOCK_SIZE = 8;
    int SAD_NEIGHBORHOOD = 16;

    for (int j = 0; j < width_1 - BLOCK_SIZE; j += BLOCK_SIZE)
    {
        for (int i = 0; i < height_1 - BLOCK_SIZE; i += BLOCK_SIZE)
        {
            a.x = j;
            a.y = i;
            int dif = std::numeric_limits<int>::max();
            int temp;

            // если SAD == 0, то изображение в этом блоке не изменилось,
иначе рисуем вектор смещения
            if (SAD(mat1(cv::Rect(j, i, BLOCK_SIZE, BLOCK_SIZE)),
mat2(cv::Rect(j, i, BLOCK_SIZE, BLOCK_SIZE))) == 0)
            {
                b.x = j;
                b.y = i;
                dif = 0;
            }
            else
            {
                int c = borders(i - SAD_NEIGHBORHOOD, height2);
                int d = borders(i + SAD_NEIGHBORHOOD, height2 - BLOCK_SIZE);
                int n = borders(j - SAD_NEIGHBORHOOD, width2);
                int m = borders(j + SAD_NEIGHBORHOOD, width2 - BLOCK_SIZE);
                for (int l = n; l <= m; l += 1)
                {
                    for (int k = c; k <= d; k += 1)
                    {
                        temp = SAD(mat1(cv::Rect(j, i, BLOCK_SIZE,
BLOCK_SIZE)), mat2(cv::Rect(l, k, BLOCK_SIZE, BLOCK_SIZE)));
                        if (temp < dif)
                        {
                            dif = temp;
                            b.x = l;
                            b.y = k;
                        }
                    }
                }
                cv::arrowedLine(mat1, a, b, CV_RGB(0, 0, 255), 1, CV_AA, 0,
0.6);
            }
        }
    }
    cv::imshow("RESULT", mat1);
}
```