Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Лабораторная работа №1 (Построение векторов смещения)**

**Дисциплина**: Разработка графических приложений

Выполнил студент гр. 13541/2 Коренёк Г.А.

(подпись)

Руководитель Болсуновская М.В.

(подпись)

“ ” 2018 г.

Санкт – Петербург

**2018**

**1. Подготовка тестовых изображений**

В качестве тестовых изображений были выбраны 2 последовательных кадра 512х512 пикселей из видео со статичным фоном и движущимся объектом.

Рис. 1.1. Тестовые изображения.

Для упрощения реализации алгоритма изображения преобразуются в черно-белый формат.

**2. Подготовка отладочных изображений**

В для отладки была создана пара из изображения с белым шумом и его копии, сдвинутой вправо на 16 пикселей.

**3. Вычисление векторов смещения**

Был использован размер блока 16х16 пикселей. В качестве метрики использовалась SAD с размером области поиска 32 блока.

Код для вычисления и визуализации векторов смещения представлен в листинге 1. Для отладочной пары вектора смещения определены верно (направление справа налево, рис. 3.1.).



Рис. 3.1. Вектора смещения для отладочной пары изображений.

Результат для тестовых изображений представлен на рис. 3.2.



Рис. 3.1. Вектора смещения для отладочной пары изображений.

Листинг 1. Построение и визуализация векторов смещения

**#include <stdio.h>**

**#include <iostream>**

**#include <algorithm>**

**#include <cmath>**

**#include <unistd.h>**

**#include <opencv2/opencv.hpp>**

**using namespace cv;**

**struct BlockCoords {**

**int row;**

**int column;**

**BlockCoords() {};**

**BlockCoords(int row, int column) {**

**this->row = row;**

**this->column = column;**

**}**

**};**

**Mat get\_block(Mat image, BlockCoords coords, int block\_size) {**

**return image.rowRange(**

**coords.row \* block\_size, (coords.row + 1) \* block\_size)**

**.colRange(**

**coords.column \* block\_size, (coords.column + 1) \* block\_size);**

**}**

**Scalar sad(Mat prev, Mat curr) {**

**Mat result;**

**absdiff(prev, curr, result);**

**return sum(result);**

**}**

**BlockCoords get\_matching\_block(Mat prev, Mat curr, BlockCoords base,**

**int block\_size, int search\_area) {**

**BlockCoords matching\_block\_coords = base;**

**int min\_sad = INT\_MAX;**

**Mat base\_block = get\_block(prev, base, block\_size);**

**int lbound\_row = max(base.row - search\_area, 0);**

**int lbound\_column = max(base.column - search\_area, 0);**

**int hbound\_row = min(base.row + search\_area, (prev.rows - 1) / block\_size);**

**int hbound\_column = min(base.column + search\_area, (prev.cols - 1) / block\_size);**

**for (int row = lbound\_row; row <= hbound\_row; row++) {**

**for (int column = lbound\_column; column <= hbound\_column; column++) {**

**BlockCoords curr\_coords(row, column);**

**Mat curr\_block = get\_block(curr, curr\_coords, block\_size);**

**int curr\_sad = sad(base\_block, curr\_block)[0];**

**if (curr\_sad < min\_sad) {**

**matching\_block\_coords = curr\_coords;**

**min\_sad = curr\_sad;**

**}**

**}**

**}**

**return matching\_block\_coords;**

**}**

**int main(int argc, char\*\* argv) {**

**if (argc != 3) {**

**printf("usage: DisplayImage.out <Image\_Path>\n");**

**return -1;**

**}**

**Mat prev = imread(argv[1], 1);**

**Mat curr = imread(argv[2], 1);**

**if (!(prev.data && curr.data)) {**

**printf("No image data \n");**

**return -1;**

**}**

**cvtColor(prev, prev, COLOR\_RGB2GRAY);**

**cvtColor(curr, curr, COLOR\_RGB2GRAY);**

**int block\_size = 16; // pixels**

**int search\_area = 32; // blocks**

**int img\_block\_rows = prev.rows / block\_size;**

**int img\_block\_columns = prev.cols / block\_size;**

**// #define DEBUG**

**#ifdef DEBUG**

**int image\_size = 512;**

**int shift = block\_size;**

**Mat template\_img(image\_size, image\_size + shift, CV\_8UC1);**

**randu(template\_img, Scalar(0, 0, 0), Scalar(255, 255, 255));**

**Mat prev\_test = template\_img.colRange(0, image\_size);**

**Mat curr\_test = template\_img.colRange(shift, image\_size + shift);**

**prev = prev\_test;**

**curr = curr\_test;**

**#endif**

**BlockCoords\*\* matching\_blocks = new BlockCoords\*[img\_block\_rows];**

**for (int row = 0; row < img\_block\_rows; row++) {**

**matching\_blocks[row] = new BlockCoords[img\_block\_columns];**

**}**

**for (int row = 0; row < img\_block\_rows; row++) {**

**for (int column = 0; column < img\_block\_columns; column++) {**

**BlockCoords curr\_coords(row, column);**

**matching\_blocks[row][column] = get\_matching\_block(**

**prev, curr, curr\_coords, block\_size, search\_area);**

**}**

**}**

**Mat result = prev;**

**for (int row = 0; row < img\_block\_rows; row++) {**

**for (int column = 0; column < img\_block\_columns; column++) {**

**int arrow\_radius = (block\_size - 1) / 2;**

**int arrow\_center\_row = row\*block\_size + arrow\_radius;**

**int arrow\_center\_column = column\*block\_size + arrow\_radius;**

**float dy = matching\_blocks[row][column].row - row;**

**float dx = matching\_blocks[row][column].column - column;**

**std::cout << dx << " " << dy << std::endl;**

**if (dx != 0) {**

**dx = dx / pow((pow(dx, 2) + pow(dy, 2)), 0.5) \* arrow\_radius;**

**}**

**if (dx != 0) {**

**dy = dy / pow((pow(dx, 2) + pow(dy, 2)), 0.5) \* arrow\_radius;**

**}**

**Point arrow\_start(-dx + arrow\_center\_column, -dy + arrow\_center\_row);**

**Point arrow\_end(dx + arrow\_center\_column, dy + arrow\_center\_row);**

**arrowedLine(result, arrow\_start, arrow\_end, Scalar(255,255,255), 1);**

**}**

**}**

**namedWindow("Display Image", WINDOW\_AUTOSIZE );**

**imshow("Display Image", result);**

**waitKey(0);**

**return 0;**

**}**