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 Балсутьев В.А.

(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 8x8 пикселей
3. Поиск похожих блоков первого изображения во втором с помощью метрики SAD в окрестности 16х16 пикселей методом полного перебора
4. Визуализация векторов смещения, наложенных на исходное изображение

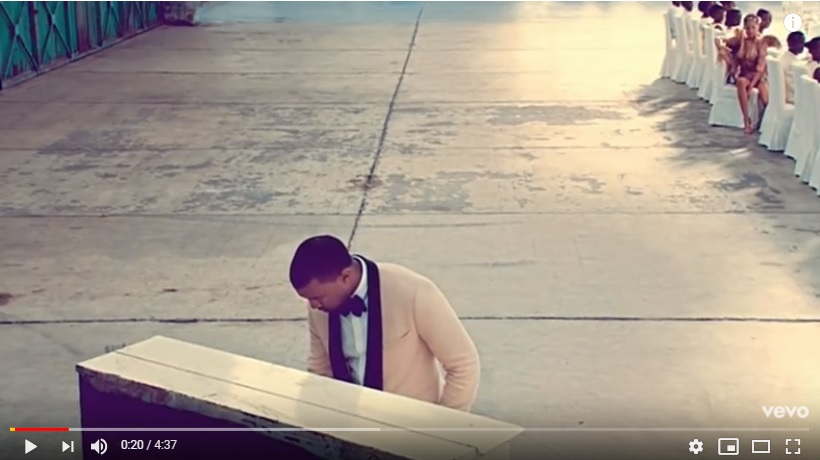
# Эксперименты:

* Нахождение и отображение векторов смещения на изображениях:
* Два последовательных кадра из нескольких роликов
* Смещение изображение вправо

# Результаты

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

|  |
| --- |
| C:\Users\Vovasq\Downloads\Telegram Desktop\Project1\Project1\kanye01.jpg |

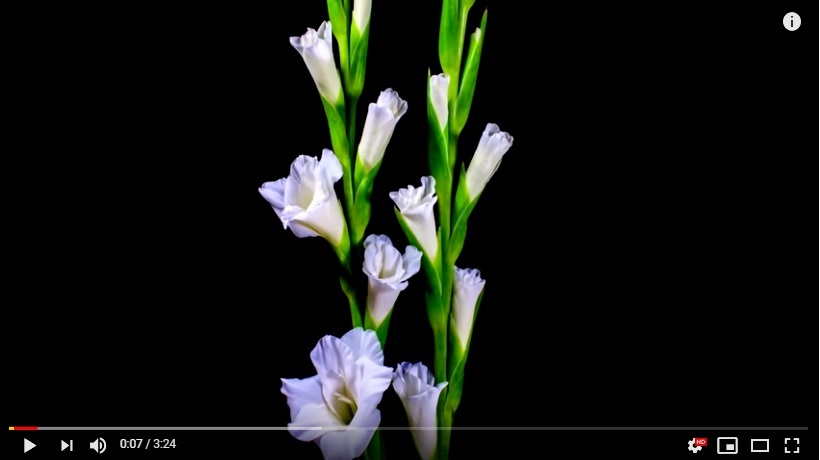


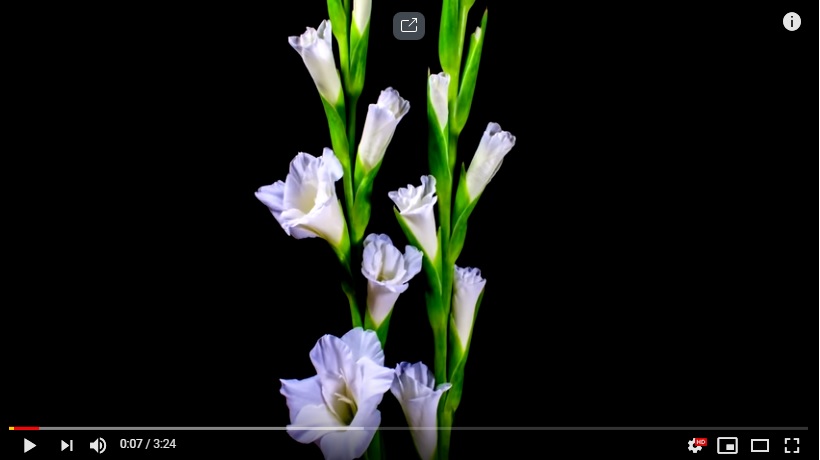
|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

В качестве результата получаем, изображение с отображением рассчитанных векторов смещения. В связи с тем, что изображения получено из клипа, где камера смещается, свет падающий на пол изменяется. Как видно из результата в верхнем правом углу видим смещение, однако фактически человеческому глазу кажется, что изменений нет.

|  |
| --- |
|  |

Повторим эксперимент, но уже с другими последовательными изображениями источника также используется видеоролик:





В результате получаем следующее изображение:



Невооруженным глазом видно, что алгоритм справился неплохо, однако, следует отметить, что при однородном цвете алгоритм не справляется.

Теперь рассмотрим изображение со сдвигом.

В данном эксперименте, проведем сдвиг изображения вправо.

Исходное изображение:



|  |
| --- |
|  |
| Изображение со сдвигом вправо  C:\Users\Vovasq\Downloads\Telegram Desktop\Project1\Project1\hol02.jpg |

Получаем в целом верный результат, однако снова заметно, что в некоторых местах

изображения алгоритм ошибается, это также связано с одинаковым цветом пикселей:

|  |
| --- |
|  |

# 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, именно под эту ситуацию подходят в частности границы.

# Приложение

Код программы

#include "opencv2/core.hpp"

#include "opencv2/imgproc.hpp"

#include "opencv2/highgui.hpp"

#include <vector>

#include <stdio.h>

#include <iostream>

int borders(int a, int b)

{

return std::max(std::min(a, b), 0);

}

int SAD(cv::Mat mat1, cv::Mat mat2)

{

int sum = 0;

for (int i = 0; i < mat1.size().height; i++)

{

for (int j = 0; j < mat1.size().width; j++)

{

for (int k = 0; k < 3; k++)

{

sum += abs(mat1.at<cv::Vec3b>(i, j)[k] - mat2.at<cv::Vec3b>(i, j)[k]);

}

}

}

return sum;

}

void ProccesImages(cv::Mat mat1, cv::Mat mat2)

{

int height1 = mat1.size().height;

std::cout << height1 << std::endl;

int width1 = mat1.size().width;

std::cout << width1 << std::endl;

int height2 = mat2.size().height;

int width2 = mat2.size().width;

CvPoint a;

CvPoint b;

for (int j = 0; j < width1 - 8; j += 8)

{

for (int i = 0; i < height1 - 8; i += 8)

{

a.x = j;

a.y = i;

int dif = std::numeric\_limits<int>::max();

int temp;

if (SAD(mat1(cv::Rect(j, i, 8, 8)), mat2(cv::Rect(j, i, 8, 8))) == 0)

{

b.x = j;

b.y = i;

dif = 0;

}

else

{

int c = borders(i - 16, height2);

int d = borders(i + 16, height2 - 8);

int n = borders(j - 16, width2);

int m = borders(j + 16, width2 - 8);

for (int l = n; l <= m; l += 1)

{

for (int k = c; k <= d; k += 1)

{

temp = SAD(mat1(cv::Rect(j, i, 8, 8)), mat2(cv::Rect(l, k, 8, 8)));

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("Changed photo", mat1);

}

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

{

cv::Mat mat1, mat2;

mat1 = cv::imread("hol01.jpg", 1);

mat2 = cv::imread("hol02.jpg", 1);

cv::imshow("Photo 1", mat1);

cv::imshow("Photo 2", mat2);

ProccesImages(mat1, mat2);

cv::waitKey(0);

return 0;

}