Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Лабораторная работа №4 (Определение векторов смещения)**

**Дисциплина**: Методы и средства цифровой обработки

Выполнил студент гр. 13541/1 Смирнов М.И.

(подпись)

Руководитель Абрамов Н.А.

(подпись)

“\_\_\_” \_\_\_\_\_\_\_\_ 2017 г.

Санкт – Петербург

2017

**Содержание**

[1. Задание 3](#_Toc504444061)

[2. Решение 4](#_Toc504444062)

[3. Вывод 13](#_Toc504444063)

[Список литературы 14](#_Toc504444064)

1. **Задание**

В данной работе требуется взять 2 последовательных кадра из какого-либо видеоролика. Найти вектор смещения для каждого блока на кадре t-1 относительно кадра t (метрики SAD и SSD). И визуализировать результат, а именно попробовать вернуть кадр обратно (восстановить из изображения t-1 изображение t).

1. **Решение**

Для решения поставленной задачи была выбрана среда Matlab.

**Этапы выполнения**:

1. Получение последовательных кадров из ролика.

Был скачан ролик из интернета с расширением .mp4 и разрешением 320х480. Было выбрано такое низкое разрешение для ускорения работы программы, ведь чтобы обойти все пиксели блоками потребуется немало времени. Далее записываем в переменную ролик (с помощью “VideoReader”) и берем оттуда 2 последовательных кадра (с помощью “read”). На самом деле лучше взять кадр через один, чтобы разница между изображениями была более заметна.

Код данной операции представлен ниже:

% Берем из ролика 2 последовательных кадра

myVideo = VideoReader('D:/Spasti\_kota.mp4');

cadre1 = 1860;

cadre2 = 1862;

t\_1 = read(myVideo,cadre1);

t = read(myVideo,cadre2);

1. Преобразование в оттенки серого.

По знакомой нам схеме преобразуем полученные изображения в оттенки серого и выведем их на экран.

% Преобразуем изображения в оттенки серого

for i = 1:size(t\_1,1)

for j = 1:size(t\_1,2)

Iser1(i,j) = 0.299\*t\_1(i,j,1)+0.587\*t\_1(i,j,2)+0.114\*t\_1(i,j,3);

Iser2(i,j) = 0.299\*t(i,j,1)+0.587\*t(i,j,2)+0.114\*t(i,j,3);

end

end

% Выводим изображения t\_1 и t

figure, imshow(Iser1);

title ('Кадр t-1 в оттенках серого');

figure, imshow(Iser2);

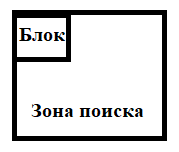
title ('Кадр t в оттенках серого');

Полученные результаты (t-1 слева, t справа):



1. Задаем начальные данные.

В работе мы стремимся, зная кадр t, восстановить кадр t-1. Для этого нам необходимо понять, что же изменилось на кадре t относительно предыдущего кадра. Т.е. мы должны пройтись по определенным блокам изображения и узнать какие смещения произошли в этих блоках. Поэтому мы задаем начальные данные, а именно: размер зоны поиска, размер блока (внутри зоны поиска) и размер шага (расстояние между блоками).



Код программы представлен ниже:

% Определяем размеры кадра

[sizeX, sizeY, number] = size(Iser1);

% Заводим входные параметры

sizeBlock = 16; % Размер блока

sizeWindow = 48; % Размер окна поиска

stride = 16; % Шаг до следующего блока

step = 1; % Текущий шаг

1. Подготовка и выбор метрики.

Далее нам требуется построить график векторов смещения. Для этого нам необходимо пройтись по всем блокам и зонам поиска вдоль всего изображения и воспользоваться одной из метрик:

* SAD:
* SSD:

В данной работе я буду применять именно метрику SAD. Это далеко не просто. Потребуется выполнить ещё множество пунктов, причем проще сделать это в одном большом цикле. Начнем с того что заведем наш будущий график путем использования функции “hold on”. Эта функция позволяет нам работать в одном изображение (в нашем случае в одном графике) путем наложения друг на друга последующих графиков, созданных в цикле.

Цикл задается по px и py и проходит по блокам с шагом stride.

% Весь график векторов смещения

figure;

hold on

for px = sizeBlock/2:stride:sizeX-sizeBlock/2

for py = sizeBlock/2:stride:sizeY-sizeBlock/2

metricSAD\_min = 0;

flag = 0;

% Искомый кадр t-1

x\_center1 = px;

y\_center1 = py;

x = 1;

y = 1;

for i = px-sizeBlock/2+1:px+sizeBlock/2

for j = py-sizeBlock/2+1:py+sizeBlock/2

Bt1(x,y) = double(Iser1(i,j)); % кадр t-1

y = y+1;

end

x = x+1;

y = 1;

end

1. Определение коэффициентов для поиска допустимой зоны поиска.

Исходя из полученных данных из цикла, выбирается значение ветвлением для коэффициентов x1Coef и x2Coef.

% Определяем коэффициенты для поиска допустимого окна

% По X

if px <= sizeBlock

x1Coef = 0;

x2Coef = sizeWindow/2-sizeBlock/2;

elseif px >= sizeX-sizeBlock/2

x1Coef = sizeWindow/2-sizeBlock/2-1;

x2Coef = 0;

else

x1Coef = sizeWindow/2-sizeBlock/2-1;

x2Coef = sizeWindow/2-sizeBlock/2;

end

Также поступим и с коэффициентами для y1Coef и y2Coef:

% По Y

if py <= sizeBlock

y1Coef = 0;

y2Coef = sizeWindow/2-sizeBlock/2;

elseif py >= sizeY-sizeBlock/2

y1Coef = sizeWindow/2-sizeBlock/2-1;

y2Coef = 0;

else

y1Coef = sizeWindow/2-sizeBlock/2-1;

y2Coef = sizeWindow/2-sizeBlock/2;

end

1. Получение второго значения для формулы SAD.

Здесь необходимо найти значение Bt2 и далее подставить его в формулу SAD. Код представлен ниже:

% ОПРЕДЕЛЕНИЕ ОКНА ПОИСКА

xCenter = x\_center1;

yCenter = y\_center1;

for wx = px-x1Coef:step:px+x2Coef

for wy = py-y1Coef:step:py+y2Coef

% Возможный блок в кадре t

x\_c = wx;

y\_c = wy;

x = 1;

y = 1;

for i = wx-sizeBlock/2+1:wx+sizeBlock/2

for j = wy-sizeBlock/2+1:wy+sizeBlock/2

Bt2(x,y) = double(Iser2(i,j)); % кадр t

y = y+1;

end

x = x+1;

y = 1;

end

1. Сравнение блоков.

Мы получили необходимые нам значения для формулы SAD и теперь мы можем ими воспользоваться для сравнения наших блоков, определив минимально отличающиеся блоки (min SAD – самый похожий блок). Код ниже:

% Сравнение блоков

for i = 1:1:size(Bt1,2)

for j = 1:1:size(Bt1,1)

S(i,j) = (abs(Bt1(i,j)-Bt2(i,j)));

end

end

SAD = sum(sum(S));

if flag == 0

metricSAD\_min = SAD;

flag = 1;

end

% Определение минимально отличающегося блока

if SAD < metricSAD\_min

metricSAD\_min = SAD;

xCenter = x\_c;

yCenter = y\_c;

end

end

end

1. Строим векторы смещения.

Наконец мы можем вывести график смещения векторов первых блоков и закрыть цикл. Далее цикл будет повторяться и на график будут накладываться новые графики смещения.

% Построение графика векторов смещения

plot([x\_center1 xCenter], [y\_center1 yCenter])

title ('График векторов смещения');

vectArray(1,number)=x\_center1;

vectArray(2,number)=xCenter;

vectArray(3,number)=y\_center1;

vectArray(4,number)=yCenter;

number= number + 1;

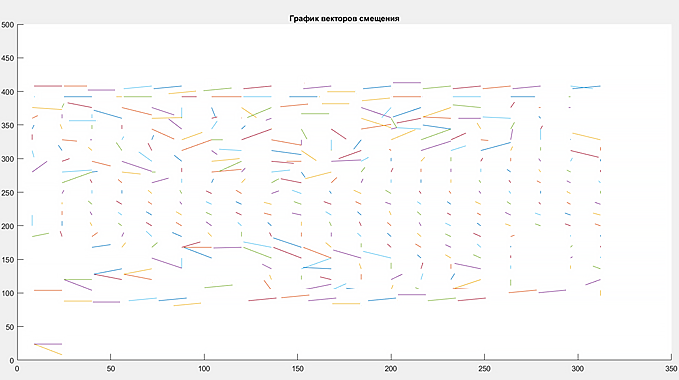
py = py - 1;

end

px = px - 1;

end

Получившийся результат:



1. Восстановление изображения.

Далее, зная кадр t и график векторов смещения, восстанавливаем кадр t-1 и записываем его в переменную ImagChange. Затем выводим его с помощью функции “imshow”. Код представлен ниже:

% ВОССТАНАВЛИВАЕМ КАДР t-1, ЗНАЯ КАДР t

for l=1:1:number-1

for i=-sizeBlock/2+1:sizeBlock/2

for j=-sizeBlock/2+1:sizeBlock/2

if ((vectArray(4,l)+j)<=sizeY)&((vectArray(3,l)+j)<=sizeY)&((vectArray(2,l)+i)<=sizeX)&&((vectArray(1,l)+i)<=sizeX)

ImagChange(vectArray(1,l)+j,vectArray(3,l)+i)=Iser2(vectArray(2,l)+j,vectArray(4,l)+i);

end

end

end

end

%Выводим получившееся изображение

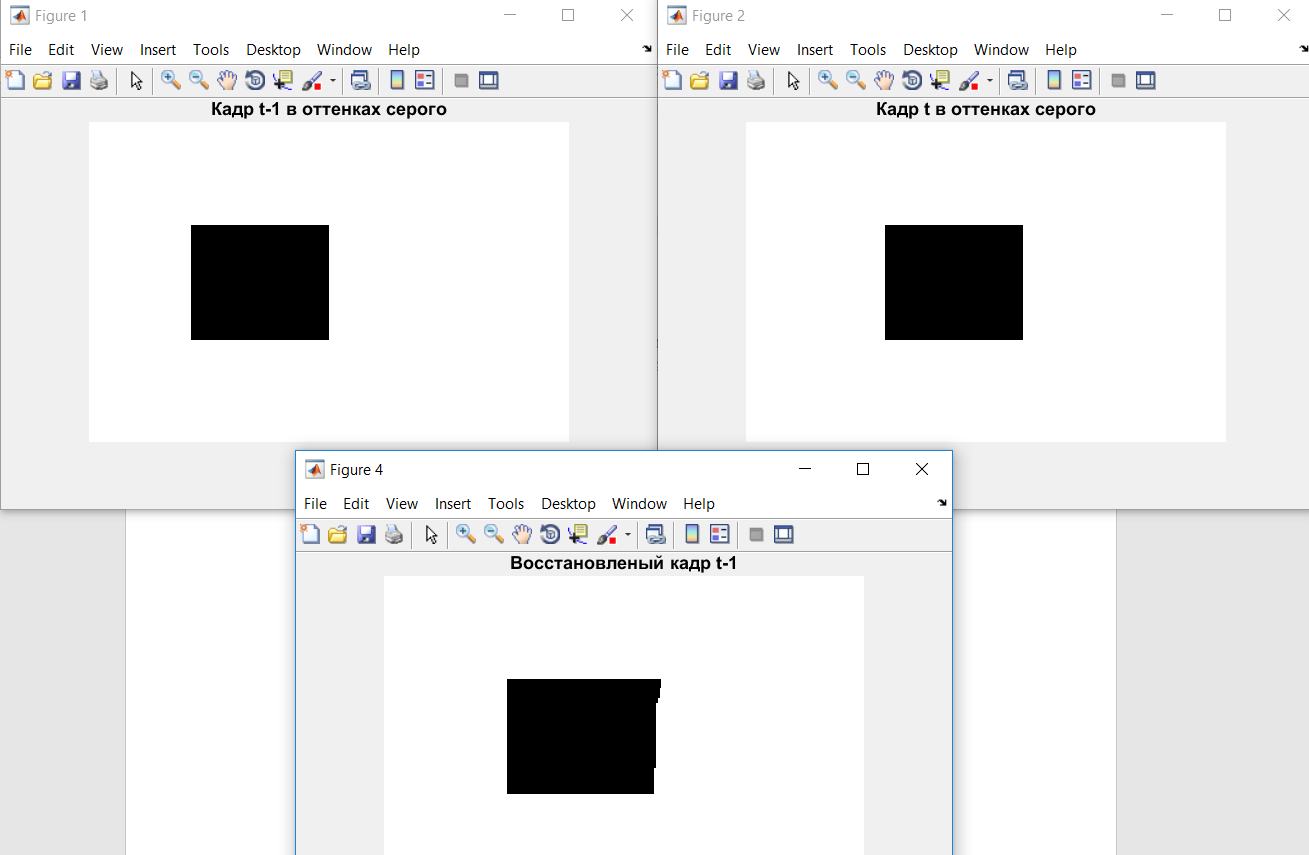
figure, imshow(ImagChange);

title ('Восстановленый кадр t-1');

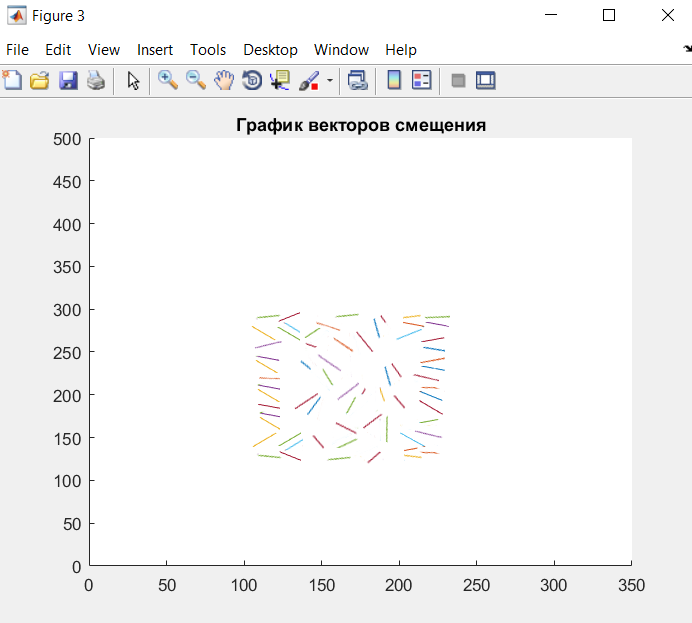
Получившийся результат:



Для проверки выполеим ту же программу для изображенний созданого нами и с известным нам смещением на последующем изображении.



Вектор смещения выглядит следующим образом:



1. **Вывод**

Компенсация движения в видеоряде является не только неотъемлемой частью повышения качества видео и алгоритмов шумоподавления, но также используется для сжатия видеоданных. Методов компенсации движения в видео существует множество. Все они используются для повышения качества видео, шумоподавления, а также сжатия. Из особенностей тех методов, которые были рассмотрены в работе можно выделить:

* Алгоритм SAD больше подходит для изображений, которые ранее прошли фильтрацию Лаплассиана, а SSD лучше использовать при гауссовском распределении.
* Изменение таких параметров как “размер блока” и “размер окна поиска” наиболее сильно влияет на качество получаемого результата для обоих методов.
* Плюс метода SAD в том, что его просто реализовать и он более точен в вычислениях, нежели остальные алгоритмы (сумма разности квадратов, нормированная кросс-корреляция).
* При использовании метода SSD лучше заметны большие различия между x и y. А также он является одним из популярнейших методов, используемых в компьютерном зрении.

**Список литературы**

1. Мир цифровой обработки. Цифровая обработка изображений. Авторы: Рафаэл С. Госалес, Ричард Е.Вудс. 2012 год.