Санкт-Петербургский государственный университет

**Кафедра теории управления**

**Леонова Екатерина Борисовна**

**Магистерская диссертация**

**Цифровая обработка изображений на основе дискретных систем**

Направление 01.04.02 - «Прикладная математика и информатика»

Магистерская программа

«Прикладная математика и информатика в задачах медицинской диагностики»

Научный руководитель,  
доктор физ.-мат. наук,  
профессор  
Котина Е. Д.

Санкт-Петербург

2019

Содержание

[Введение 3](#_Toc9425758)

[Постановка задачи 10](#_Toc9425759)

[Вариация функционала 13](#_Toc9425760)

[Алгоритм 16](#_Toc9425761)

[Блок-схема алгоритма 20](#_Toc9425762)

[Реализация алгоритма 24](#_Toc9425763)

[Заключение 29](#_Toc9425764)

[Список литературы 30](#_Toc9425765)

[Приложение 34](#_Toc9425766)

Введение

Благодаря развитию технологий человечество стало способным получать изображения не просто «копируя действительность» с помощью фотоаппарата, но и «реконструируя» изображение из информации, полученной с помощью современной техники: компьютерные телескопы, электронные микроскопы, компьютерные томографы и т.д. Но очень часто не достаточно просто получить картинку, необходимо также произвести дальнейшие манипуляции над изображением, чтобы добиться поставленной перед исследователем цели. Таким образом, всё большее распространение и развитие получают различные методы обработки изображений.

На выходе, в результате обработки изображений, мы можем получать не только модифицированное изображение, как в случае ретуши фотографий, подготовки к полиграфическому тиражированию, коррекции видеоряда. Но и различного рода информацию, которая применяется для решения множества разнообразных прикладных и научных задач: контроль популяции диких животных при помощи аэросъёмки, распознавание рукописного текста, автопилотирование машин и роботов, определение типа клеток и их подсчёт с помощью электронного микроскопа, создание 3D изображений и многое другое.

Условно обработку изображений можно разделить на два направления:

* Обработка статичных изображений;
* Обработка видео (последовательности кадров).

Как в первом, так и во втором случае может рассматриваться последовательность изображений, но если исследователей интересует содержание изображений, то будем говорить об обработке каждого изображения в отдельности. Примером результатов такой обработки может быть классификация объектов на изображении (корреляция изображения или отдельных объектов на нём с заданными образцами), выделение контуров объектов, и т.п.

Обработкой же последовательности кадров будем называть ситуации, когда для решения поставленных задач необходимо получить информацию об изменениях, что происходят при переходе от одного кадра к другому. В данном случае, чаще всего, говорят об идентификации направления и скорости перемещения объекта на изображениях.

Информация подобного рода необходима, например, при решении задач управления передвижением беспилотных машин и роботов. На её основе могут делаться выводы о том, насколько близко к камере находится объект, как он взаимодействует с другими объектами на изображении, т.е. благодаря этим данным машина может получить информацию о пространственной структуре окружающей среды и своём положении в ней и, таким образом, ориентироваться в пространстве вокруг.

Также, зная направление и скорость перемещения объекта на последовательности изображений, можно производить автоматизацию корректировки положения интересующей нас области на картинках. Данная операция необходима, когда для дальнейшей обработки изображений требуется сохранение пространственного положения объекта, а при создании изображений по тем или иным причинам сохранение статичного положения объекта невозможно. Например, при проведении радионуклидного исследования, которое занимает достаточно длительный период времени, возможны непроизвольные смещения пациента или же его внутренних органов, что приводит к ошибкам при дальнейшей обработке этой последовательности изображений с целью оценки функционального состояния органов человека.

При использовании различных методов обработки последовательности кадров с целью определения смещения объектов на выходе получают результаты, представленные в двух видах: поле скоростей или поле перемещений. В обоих случаях мы имеем поле векторов, которое представляет собой область, размером соответствующую рассматриваемому изображению, на которой изображены вектора. В первом случае, они указывают направление и скорость смещения каждого пикселя (или области пикселей). Во втором случае, они указывают, куда именно переместился объект (пиксель), т.е. вектор соединяет точки его начального и конечного положения в результате перемещения. При этом, зная частоту дискретизации по кадрам, можно преобразовать поле скоростей в поле векторов и наоборот.

Метод оптического потока (ОП) является одним из наиболее распространенных подходов к выделению движущихся объектов на изображениях. Он предоставляет достаточно эффективный и гибкий аппарат для анализа смещений объектов на последовательностях изображений.

В методе ОП [8], [9], [10] изначально предполагается, что изображаемая поверхность плоская, и что освещение равномерно по всей поверхности объекта. Яркость в точке изображения пропорциональна отражению поверхности в соответствующей точке объекта. Кроме того, предполагается, что отражение меняется плавно и не имеет пространственных разрывов. Это последнее условие гарантирует нам, что яркость изображения является дифференцируемой.

Это подводит нас к главному предположению ОП, которое говорит о том, что яркость конкретной точки при перемещении остаётся неизменной. Пусть – это яркость изображения в точке в момент времени . Перемещение описывается следующей системой дифференциальных уравнений:

(1)

Таким образом, уравнение, связывающее изменения яркости изображения в точке с движением яркостного образа, выглядит следующем образом:

(2)

Конечно, нельзя гарантировать выполнение данного условия в полной мере, потому что со временем источник освещения может менять своё положение в пространстве, может появиться дополнительный источник света и т.п. Но экспериментально показано, что такое предположение работает довольно хорошо, благодаря чему уже не одно десятилетие исследователи имеют возможность использовать его для определения ОП.

Из формулы (2) получаем формулу (3):

(3)

Теперь мы имеем одно линейное уравнение с двумя неизвестными :

(4)

где – это обозначения для частных производных яркости изображения по соответственно.

Полученное уравнение содержит две неизвестные и не может быть однозначно разрешено. Следовательно, необходимо ввести дополнительное ограничение. С этого момента происходит деление методов нахождения ОП на разнообразные подходы, среди которых особое место занимают метод Лукаса-Канаде (Lukasa-Kanade) [8] и метод Хорна-Шанка (Horn-Schunck) [9]. Оба метода по-прежнему не теряют своей актуальности, и на работы их создателей по-прежнему ссылаются исследователи, занимающиеся вопросом ОП. Первый метод относится к локальным – для вычисления и используются значения в соседних точках некоторой окрестности. Полученные и характеризуют всю область, для которой они были вычислены (блок пикселей). А метод Хорна-Шанка относится к глобальным методам, который позволяет вычислять и для каждого отдельного пикселя изображения.

К дифференциальным методам также относится работа Нагеля. Нагель был одним из первых, кто использовал производные второго порядка для измерения оптического потока [11,12]. Также как и в методе Хорна-Шанка, в [11,12]] используется глобальное ограничение гладкости.

Методы регионального сопоставления [13] определяют скорость как сдвиг, который дает наилучшее соответствие между областями изображения в разное время. Поиск наилучшего соответствия означает максимизацию меры сходства, такой как нормализованная перекрестная корреляция или минимизация меры расстояния, например, сумма квадратов разности (SSD).

Метод сопоставления Анандан (Anandan) [14], основан на пирамиде Лапласа и стратегии сопоставления крупных, а затем и более мелких смещений с помощью SSD. Пирамида Лапласа [15] позволяет вычислять большие перемещения между кадрами и помогает улучшить структуру изображения, что часто считается важным. Затем Анандан использует ограничение гладкости для результирующих оценок скорости.

Синг (Singh) [16] предлагает стратегию SSD и использует пирамиду Лапласа, как в [14]. Это способствует уменьшению требуемых вычислительных мощностей. Наконец, Сингх предлагает собственные значения обратной ковариационной матрицы в качестве меры доверия.

Следующий класс методов оптического потока основан на выходной энергии фильтров, настроенных на скорость [1,17-22]. Они также называются частотными методами, так как используемые фильтры основаны на преобразовании Фурье [23-25]. Интересно, что было показано, что некоторые энергетические методы эквивалентны корреляционным методам и дифференциальному подходу Лукаса - Канаде [25].

Метод, разработанный Хигером [26], сформулирован как нахождение близкой к некоторому значению пространственно-временной энергии в частотном пространстве с помощью метода наименьших квадратов. Локальная энергия извлекается с помощью Габор (Gabor) фильтров, настроенными на различные пространственные ориентации и различные временные частоты. Эти фильтры применяются к каждому уровню пирамиды Гаусса.

Фазовые методы определяют скорость в терминах фазового состояния выходов полосового фильтра. К этим методам относятся методы нулевого пересечения (zero-crossing) [27], поскольку нулевые пересечения можно рассматривать как горизонтальные фазовые пересечения.

Обобщенное использование фазовой информации для оптического потока было впервые разработано Флитом и Джепсоном [28]. Данный метод определяет скорость в терминах мгновенного движения, перпендикулярного горизонтально-фазовым контурам на выходе полосовых фильтров, настроенных на скорость. Полосовые фильтры используются для разложения входного сигнала в соответствии со значением скорости и её направлением.

Ваксман, Ву и Бергхольм [29] применяют пространственно-временные фильтры к бинарным краевым картам для отслеживания краев в реальном времени.

В связи с активным развитием нейронных сетей также широкое распространение в задачах детектирования движений на последовательности изображений в последние годы получили нейронные сети [30-34]. Интересно, что обучение таких сетей рекомендуется проводить с помощью синтетических изображений [35].

При проведении динамических исследований в радионуклидной диагностике [2] информацию записывают непрерывно или через короткие промежутки времени и отображают на целой серии кадров. Интервалы между кадрами выбирают с учетом скорости изучаемых биологических процессов. С помощью полученной на выходе последовательности кадров, происходит функциональная оценка работы внутренних органов человека. Для этого, в том числе, широко используются различные математические методы, на корректность работы которых могут повлиять различные непроизвольные смещения пациента или его внутренних органов.

При написании данной работы перед нами ставилась задача по детектированию подобных смещений (определению поля скоростей) при обработке последовательных изображений на основе дискретных систем. Для успешного выполнения данной задачи перед нами ставились следующие цели:

* предложить математический метод для обработки изображений, с целью определения поля скоростей;
* разработать алгоритм определения поля скоростей на основе предложенного метода;
* написать программу, реализующую разработанный алгоритм;
* провести экспериментальную апробацию разработанной программы.

Постановка задачи

Рассмотрим дискретную систему вида:

где – -мерная векторная функция, которая определена и непрерывна на множестве по всем своим аргументам при каждом . Здесь – это некоторая область в , а является компактным множеством в где . Частные производные и вторые частные производные также определены и непрерывны по этим аргументам на множестве . Здесь – -мерный вектор состояния, характеризующий изменение координат точек изображения в последовательности кадров, – -мерный вектор.

Также считаем, что при любых изменениях якобиан

= не равен нулю.

Полагаем, что уравнение (5) описывает перемещение (изменение координат) точек изображения в последовательности кадров.

Обозначим яркость изображения на -ом шаге (или функцию плотности распределения радиофармпрепарата в случае радионуклидного изображения [2]). Здесь и далее будем также использовать обозначение

Положим, что начальное состояние системы (5) описывается компактным в множеством ненулевой меры с плотностью распределения (яркостью) . Последовательность векторов обозначим как . Последовательность векторов будем называть траекторией движения и обозначать через Тогда с помощью обозначим состояние траектории на -ом шаге.

Ансамблем траекторий будем называть множество траекторий , которые соответствуют различным начальным состояниям и .

Сечением ансамбля траекторий будем называть состояние ансамбля траекторий на –ом шаге, которое обозначим как :

Рассмотрим, каким образом вдоль траектории будет изменяться яркость (плотность) Фиксируем шаг и точку Тогда вследствие системы (5) является образом . Исходя на -ом шаге из , на -м шаге траектории системы оказываются в некоторой -окрестности точки Обозначим множество таких точек через

Предел вида (6) описывает плотность распределения траектории системы (1) в точке [4]:

где

Далее выведем уравнение, которому бы удовлетворяла функция яркости (плотности) на -м шаге учитывая, что преобразование (5) взаимно однозначное. Для этого произведём замену переменных вида в (6). В данном случае Якобиан преобразования принимает вид следующего выражения

Вследствие взаимно однозначного соответствия множеств и интеграл под знаком предела преобразуется к виду

Здесь и далее

Вследствие чего можно записать уравнение вида (8) для

Введём следующий функционал качества, который позволит нам оценить динамику движения и провести его оптимизацию:

где функция имеет следующий вид:

Здесь – это некоторая заданная функция. Далее обозначим

Рассмотрим совместно уравнения (5) и (8):

Для данной системы нам необходимо найти такое допустимое , чтобы минимизировать функционал качества (9).

Вариация функционала

Введём следующие обозначения для вариации траекторий системы (11) при допустимой вариации и данном

Тогда система в вариациях для системы (11) будет иметь вид:

**Лемма 1.** Имеет место следующее равенство [5]:

+

.

В свою очередь, по лемме 1, справедливо равенство вида (15):

где

Далее найдём приращение функционала (9) при допустимых управлениях и которым соответствуют траектории :

где а якобиан преобразования может быть представлен в виде [3,6]:

Тогда приращение функционала (16) принимает следующий вид:

Затем, выделяя линейные члены по , и принимая во внимание, что значения имеют порядок малости более высокий, чем равномерно по и получаем

здесь

Рассмотрим только один шаг процесса , например, только два последовательных кадра, тогда система (5) имеет вид:

и получим для этого случая, в силу уравнений (12,13,15) и благодаря методике исследования функционала вида (9), представленную в работах [4,5,6] вариацию функционала (9):

Выберем такие вспомогательные функции . Обозначим Также пусть функция яркости на -ом шаге будет равна функции яркости на -ом шаге (случай оптического потока). Тогда в силу равенства (8) якобиан равен 1. В результате, учитывая начальные значения вариаций получим:

Благодаря представлению (21) вариации функционала (9) , мы имеем возможность использовать различные направленные методы оптимизации.

Алгоритм

Рассмотрим случай, когда , тогда система (11) принимает следующий вид:

где – 2-мерный вектор, а – матрица размерности 2х2, где , и – 2-мерный вектор. Соответственно, вектор искомых параметров будет состоять из компонент матрицы и вектора и иметь следующий вид .

Далее будем рассматривать случай, когда мы предполагаем, что в результате преобразования возможен только сдвиг и поворот. Тогда матрица будет иметь вид =. В этом случае вектор искомых параметров

Стоит отметить, что мы знаем фактическое значение яркости (плотности) на каждом кадре последовательности изображений, обозначим её , причем . Это объясняется тем, что данную информацию мы получаем при считывании изображений. Вместе с тем в математической модели (22-23) изменение яркости (плотности) вдоль траектории задаётся соотношением (23).

Рассмотрим задачу минимизации функционала (9). В этом случае для системы (22-23) в вариации функционала (21) принимают вид:

Получаем градиент функционала (9):

На основе полученного градиента мы можем строить направленные методы оптимизации.

Далее, для реализации нашего алгоритма введём дискретизацию по пространственным координатам. Рассмотрим функцию , при фиксированном будет функцией двух пространственных переменных и . Будем рассматривать её на ограниченной прямоугольной области изображения. Введём фиксированную сетку точек с приращением шага равным ширине одного пикселя вдоль оси ,и с приращением шага равным высоте одного пикселя вдоль оси . Размерность сетки определяется размером рассматриваемого изображения с учётом шага дискретизации в один пиксель. Решения будут находиться в узлах этой двумерной сетки, таким образом, нам будут известны значения яркости (плотности распределения) для каждого пикселя изображения , где Функция – функция двух дискретных переменных.

Для вычисления частных производных мы будем использовать значения яркости в узлах сетки и оператор Собеля [7], который основан на двумерной свёртке изображения небольшими сепарабельными целочисленными фильтрами в вертикальном и горизонтальном направлениях. Тогда . Где вычисляются следующим образом:

(26)

Ниже представлен алгоритм, отображающий последовательность операций для нахождения поля скоростей.

Нет

Да

Конец

Выводим поле скоростей

Проверка условий остановки алгоритма. Выполняется?

Вычисление частных производных

Вычисление градиента

Вычисление функционала

Входные данные изображения и начальное приближение

Начало

Блок-схема алгоритма

Рассмотренный выше алгоритм более подробно отображён с помощью блок-схемы, которая будет реализована в среде Matlab.

Начало

Задаём изображения 1 и 2

Задаём размеры блоков, для которых будет производиться расчёт ( )

Задаём и которые равны размеру изображения по оси и соответственно

:

:

Выводим массив (поле скоростей)

Направленным методом оптими- зации находим оптимизированное

Задаём

Задаём массив со значениями для каждого блока:

Конец

Более подробный разбор блока «Направленным методом оптимизации находим оптимизированное » .

Вычисляем длину шага

Вычисляем градиент , который обозначим как

Да

Да

Нет

Нет

Вычисляем значение функции

Модуль градиента = 0

Начало

Конец

Передаём на выход

Да

Нет

Вычисляем градиент

Градиентный спуск проводится по группам компонент вектора : по , и по .

Ниже представлены результаты реализации разработанного алгоритма.

Реализация алгоритма

Представленный выше алгоритм был реализован в среде Matlab. На рисунке 1 представлены результаты работы программы, где изображение делилось по горизонтали на 3 равные рассматриваемые области.

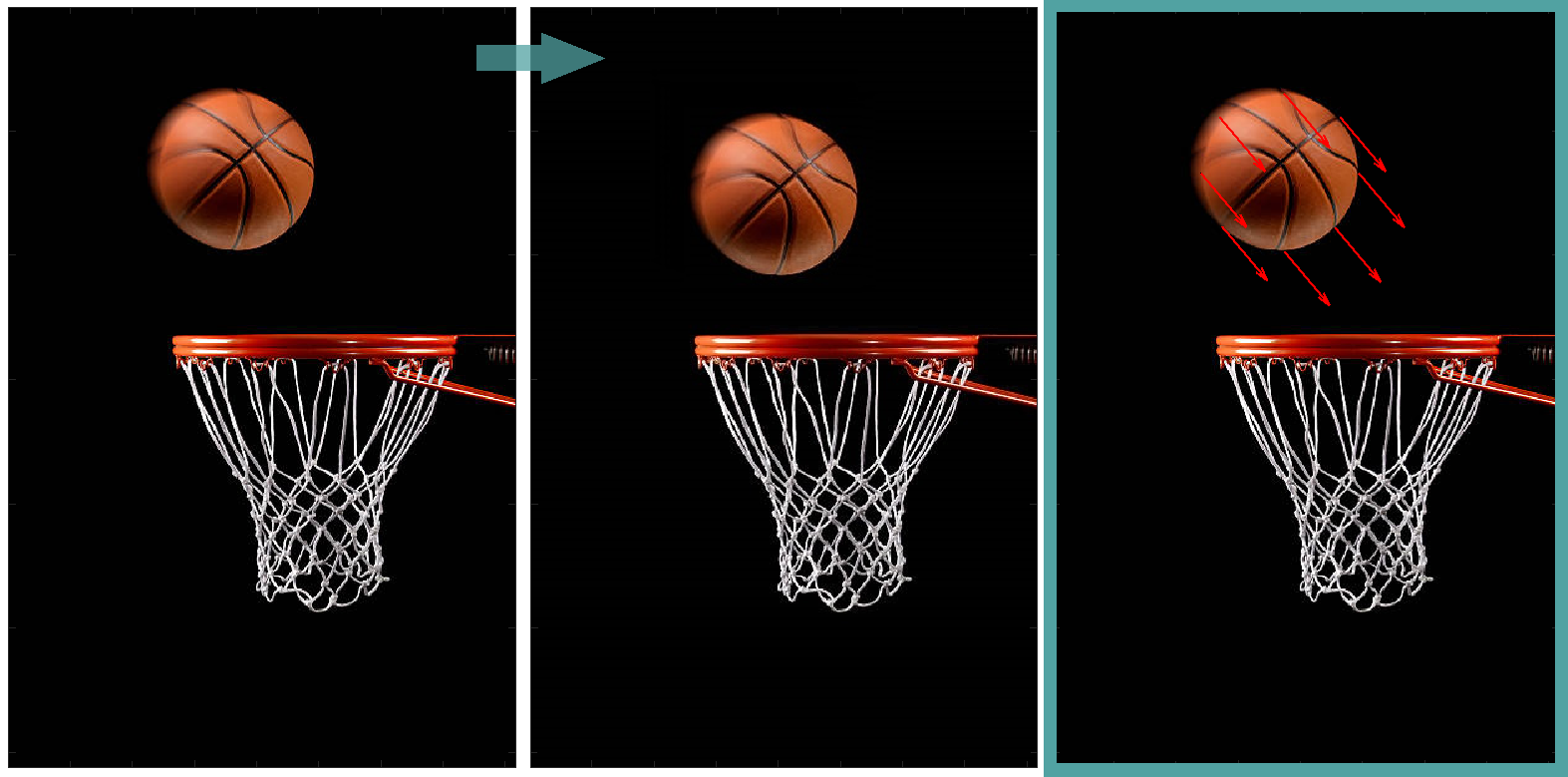


Рисунок 1 – Определения поля скоростей для различных участков изображения. Слева направо: кадр 1, кадр 2, результаты работы программы (поле скоростей)

В первой области находится движущийся баскетбольный мяч, в результате работы программы было определено направление его смещения относительно двух кадров, которое представлено справа на рисунке 1. В две другие области в разной мере попадает статичная корзина, для которой было определено смещение равное нулю.

На эти два последовательных кадра был наложен гауссовский шум с дисперсией равной 0,2. Несмотря на это, поле скоростей продолжает определяться верно, что отображено на рисунке 2.



Рисунок 2 – Результаты определения поля скоростей для изображения с шумом

На рисунке 3 представлены радионуклидные изображения достаточно «подвижного» органа - желчного пузыря. Он способен совершать за время проведения радионуклидного исследования критичные для дальнейшей оценки его функционального состояния передвижения, даже если пациент в это время остаётся неподвижным.

В данном случае, для апробации работы программы, проводилось два варианта определения поля скоростей: в первом случае, производилось разбиение изображений на области, когда смещение желчного пузыря определялось отдельно (без вклада соседних органов, которые отображаются на изображениях синим цветом), во втором – нет. В результате было установлено, что в обоих случаях в областях интереса (та часть изображения, где находится желчный пузырь) смещение определялось одинаково.

Данный метод определения поля скоростей может быть использован для корректировки смещений в радионуклидной диагностике, без необходимости делить изображение на отдельные области и задавать функцию, которая бы определяла уровень значимости таких областей, в зависимости от того, куда попадает интересующий нас орган. Это возможно благодаря тому, что яркость каждого пикселя изображения определяется накоплением радиофармпрепарата, и область нашего интереса без дополнительных манипуляций вносит максимальный вклад в функционал качества вида (9) , который нам необходимо минимизировать.

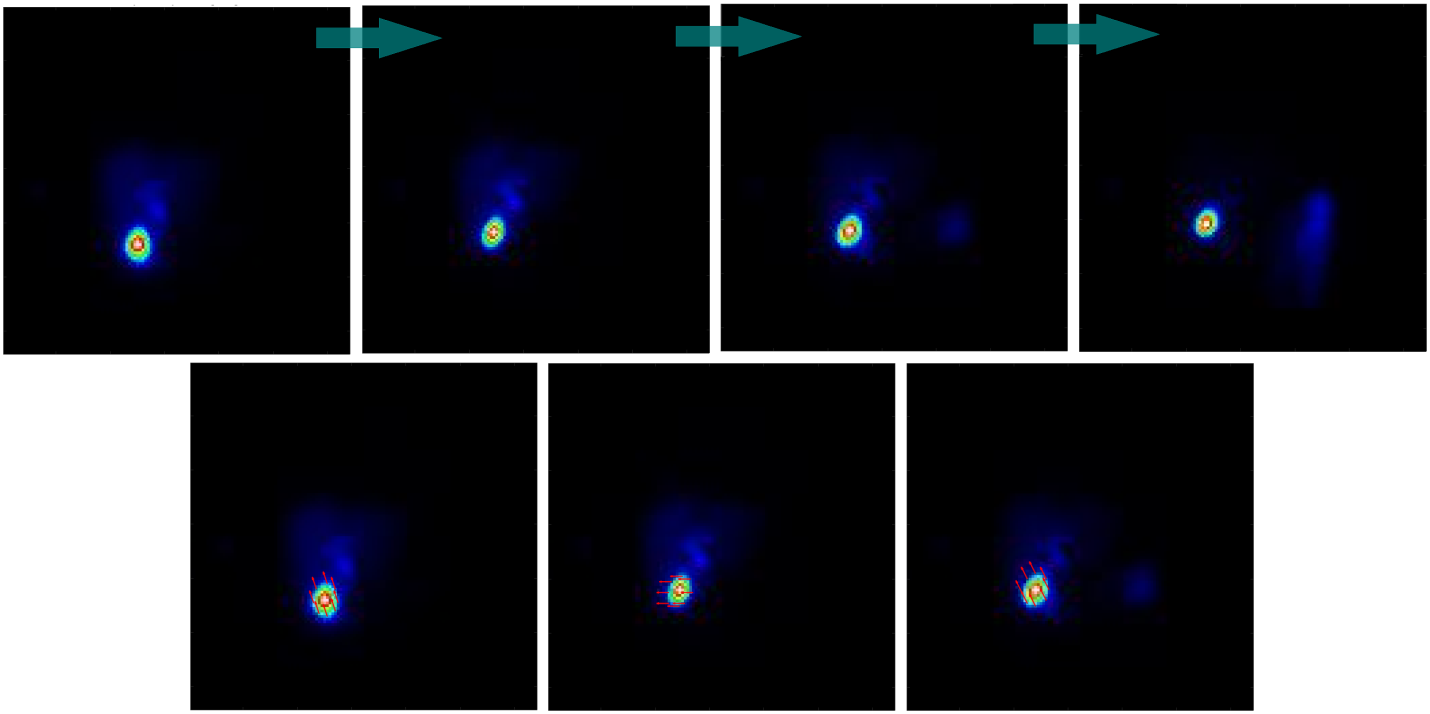


Рисунок 3 – Определение поля скоростей для желчного пузыря. Сверху – радионуклидные изображения желчного пузыря. Снизу – поле скоростей, наложенное на рассматриваемые изображения

На рисунке 4 были совмещены 2 изображения желчного пузыря: слева вверху и справа внизу находится изображение 1, а справа вверху и слева внизу – изображение 2. Для более наглядного представления смещения проведены зелёные линии. Красными векторами отображается результат работы программы (поле скоростей).

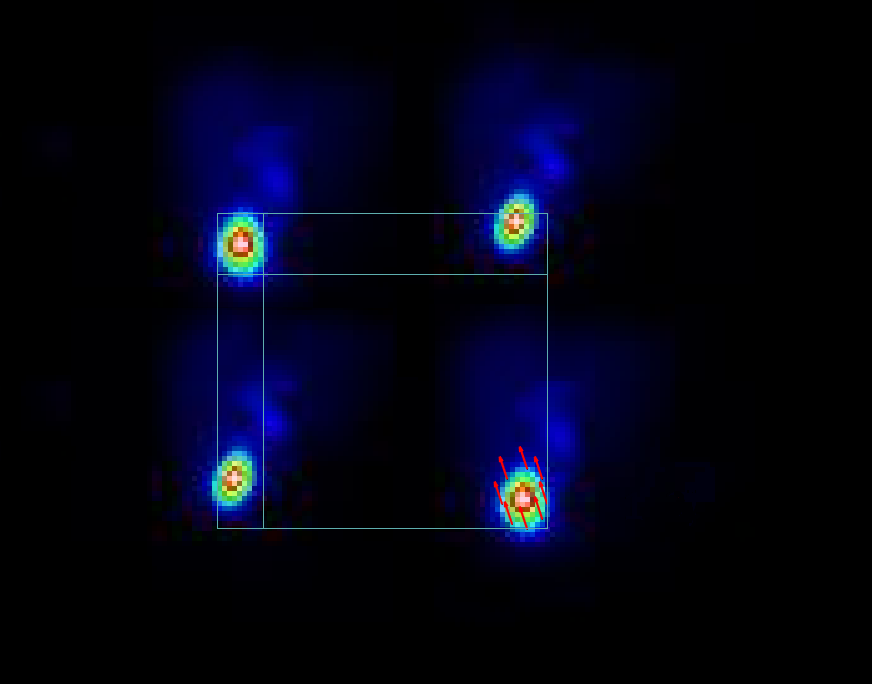


Рисунок 4 – Определения поля скоростей для двух радионуклидных изображений желчного пузыря

На рисунке 5 изображён результат определения поворота изображения. А на рисунке 6 изображён результат определения поворота изображения и его небольшого смещения по диагонали (вверх и вправо).

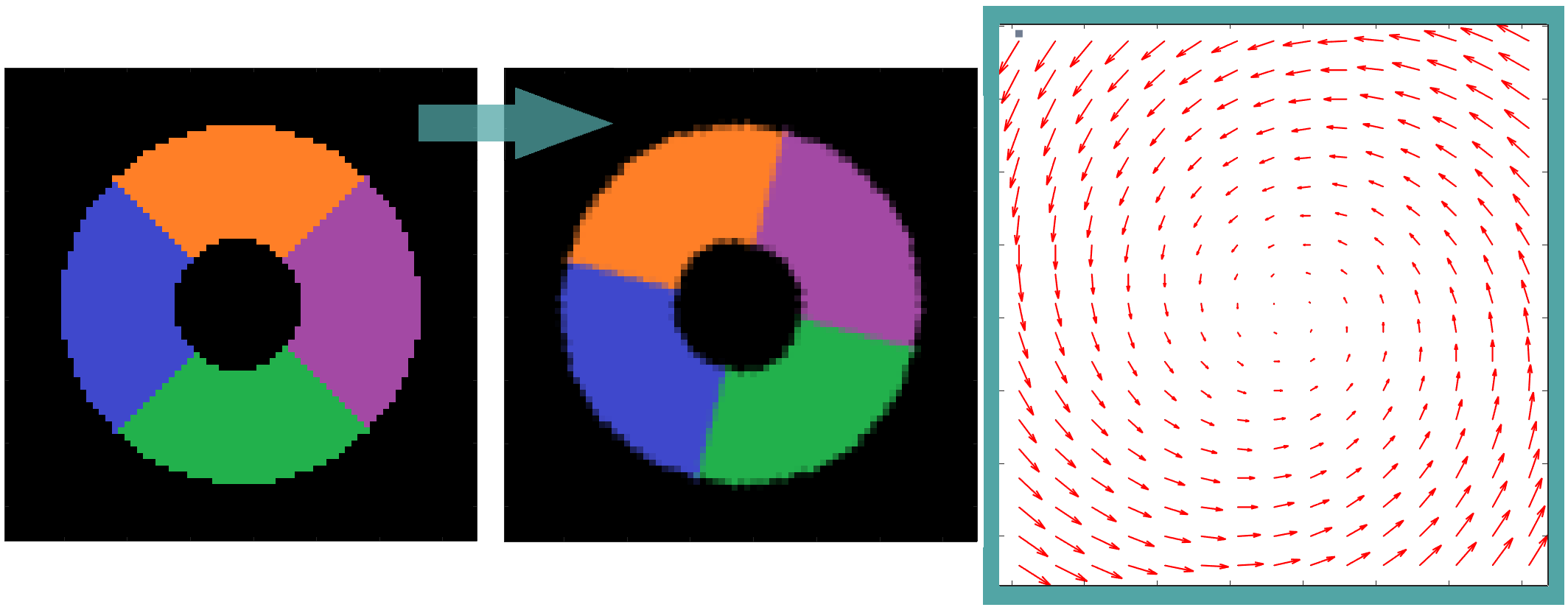


Рисунок 5 – Определение поля скоростей. Слева направо: кадр 1, кадр 2, результаты работы программы (поле скоростей)

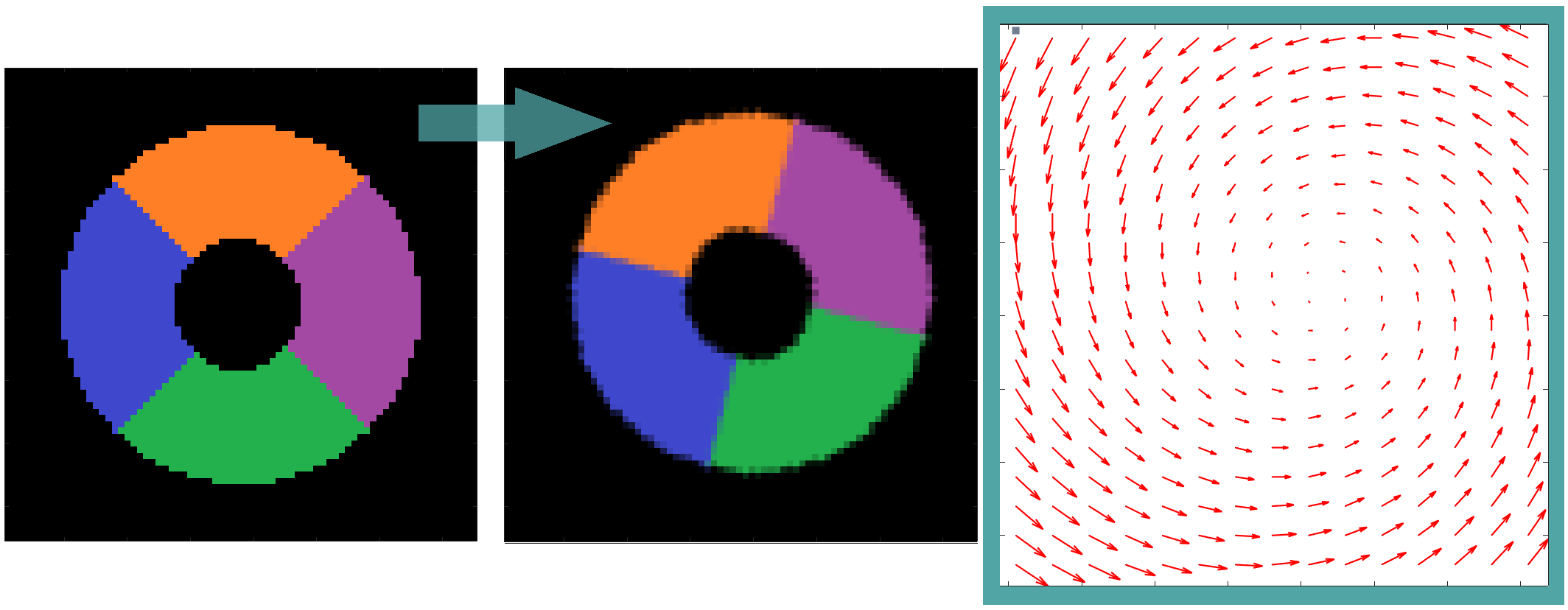


Рисунок 6 – Определение поля скоростей. Слева направо: кадр 1, кадр 2, результаты работы программы (поле скоростей)

Заключение

В результате проведённой работы был:

* предложен математический метод для обработки изображений на основе дискретной системы;
* разработан алгоритм определения поля скоростей;
* написана программа, реализующая разработанный алгоритм в среде Matlab на тестовых и радионуклидных изображениях;
* проведена экспериментальная апробация разработанной программы.

Список литературы

* + - 1. Гонсалес, Р. Цифровая обработка изображений / P. Гонсалес, Р. Вудс. –М.: Техносфера. – Издание 3-е, испр. и доп. – 2012. – 1104 с.
      2. Гребенщиков, В.В. Физико-технические основы ядерной медицины : учебное пособие / В.В. Гребенщиков, Е.Д. Котина. – СПб.: СПбГУ, 2007. – 171 с.
      3. Котина, Е.Д. Математическая модель дискретной оптимизации динамики пучка заряженных частиц / Е.Д. Котина. – СПб.: Вест. С.-Петерб. ун-та Сер. 10: Прикладная математика, информатика, процессы управления. – 2006. – Вып.2. – 30-38 с.
      4. Котина, Е.Д. Диссертация на соискание ученой степени доктора физико-математических наук / Е.Д. Котина. – СПб.: Из-во Санкт-Петербургского ун-та, 2010. – страницы.
      5. Овсянников, Д.А. Математические методы управления пучками / Д.А. Овсянников. – Л.: Изд-во Ленингр. ун-та, 1980. – 228 с.
      6. Овсянников, Д.А. Моделирование и оптимизация динамики пучков заряженных частиц / Д.А. Овсянников. – Л.: Изд-во Ленингр. ун-та, 1990. – 312 с.
      7. Расселл, Д. Оператор Собеля / Д. Расселл, Р. Кох. – VSD, 2013. – 106 с.
      8. Bruhn, A. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods / A. Bruhn, J. Weickert, C. Schnorr. // International Journal of Computer Vision. – 2005. – 61(3). – P.211 – 231.
      9. Horn, B.K. Determining optical flow / B.K. Horn, B.G. Schunck. // Artificial Intelligence. – 1981. – P.185 – 203.
      10. Fleet, D.J. Optical flow estimation / D.J. Fleet, Y. Weiss. // Mathematical Models in Computer Vision. The Handbook. – 2005. – Chapter 15. – P.239 – 258.
      11. Nagel, H.H. Displacement vectors derived from second-order intensity variations in image sequences / H.H. Nagel. // Computer Vision, Graphics, and Image Processing. – 1983. – 21(1). – P.85 – 117.
      12. Demetz, O. Feature invariance versus change estimation in variational motion estimation / O. Demetz. // Mathematische Bildverarbeitungsgruppe Fakultät für Mathematik und Informatik, Universität des Saarlandes, 2015. – P.166.
      13. Xia, Y. Dense matching comparison between census and a convolutional neural network algorithm for plant reconstruction / Y. Xia, J. Tian, P. d’Angelo, P. Reinartz. // ISPRS Journal of Photogrammetry and Remote Sensing and Spatial Information Sciences. – 2018. – P.303 – 309.
      14. Anandan, P. A computational framework and an algorithm for the measurement of visual motion. / P. Anandan. // International Journal of Computer Vision. – 1989. – 2(3). – P.283–310.
      15. Ahearn, S.C. Recursive multi-frequency segmentation of movement trajectories (ReMuS). / S.C. Ahearn, S. Dodge. // Methods in Ecology and Evolution. – 2018. – 9(4). – P.1075 – 1087.
      16. Singh, A. An estimation – theoretic framework for image flow computation. / A. Singh. // Osaka: Proc. ICCV. – 1990. – 3. – P.168 – 177.
      17. Battaglini, L. Fast random motion biases judgments of visible and occluded motion speed. / L. Battaglini, M. Maniglia, M. Konishi, G. Contemori, A. Coccaro, C. Casco, // Vision Research. – 2018. – 150. – P.38–43.
      18. Heeger, D.J. Optical flow using spatiotemporal filters. / D.J.  Heeger. // International Journal of Computer Vision. – 1988. – 1. – P.279–302.
      19. Honegger, D. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. / D. Honegger, L. Meier, P. Tanskanen, M. Pollefeys. // In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). – 2013. – P.1736–1741.
      20. Plett, J. Bioinspired visual ego-rotation sensor for mavs. / J. Plett, A. Bahl, M. Buss, K. K¨uhnlenz, A. Borst. // Biological Cybernetics. – 2012. – 106. – P.51–63.
      21. Haglund, M. Optical imaging of epileptiform and functional activity in human cerebral cortex. / M. Haglund, G.A. Ojemann, D.W. Hochman. // Nature. – 1992. –– 358(6388). – P.668–671.
      22. Von Schmude, N. Relative pose estimation from straight lines using optical flow-based line matching and parallel line clustering. / N. Von Schmude, P. Lothe, J. Witt, B. Jähne. // Computer Vision, Imaging and Computer Graphics Theory and Applications. – 2017. – 329–352.
      23. Adelson, E.H. The plenoptic function and the elements of early vision. / E.H. Adelson, J.R. Bergen. // Computational models of visual processing. Cambridge, MA: The MIT Press. – 1991. – P.3–20.
      24. Watson, A.B. Model of human visual-motion sensing. / A.B. Watson, A.J. Ahumada. // J.Opt.Soc.Am.A. – 1985. – P.322–42.
      25. Barron, J.L. Performance of optical flow techniques. / J.L. Barron, D.J Fleet, S.S. Beauchemin. // International Journal of Computer Vision. – 1994. – 12(1). – P.43–77.
      26. Heeger, D.J. Optical flow using spatiotemporal filters. / D.J. Heeger. // International Journal of Computer Vision. – 1988. – 1. – P.279–302.
      27. Hildreth, E. The computation of the velocity field. / E.  Hildreth. // Proc. [Royal Society](http://royalsocietypublishing.org/) Lond. – 1984. – 221. – P.189-220.
      28. Fleet, D.J. Computation of component image velocity from local phase information. / D.J.  Fleet, A.D. Jepson. // International Journal of Computer Vision. – 1990. – 5(1) . – P.77–104.
      29. Waxman, A.M. Convected activation profiles and the measurement of visual motion. / A.M.  Waxman, J. Wu, F. Bergholm. // CVPR, The Computer Society Conference on Computer Vision and Pattern Recognition. – 1988. – P.17-23.
      30. Doerig, A. Building perception block by block: a response to Fekete et al. / A.  Doerig, F. Scharnowski, M. Herzog. // Neuroscience of Consciousness. – 2019(1) – URL: https://doi.org/10.1093/nc/niy012.
      31. Sun, D. CNNs for optical flow using pyramid, warping, and cost volume. / D. Sun, X. Yang, M. Liu, J. Kautz. // PWC-Net: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2018. – P.8934-8943.
      32. McNally, W. Action recognition using deep convolutional neural networks and compressed spatio-temporal pose encodings / W. McNally, A. Wong, J. McPhee. // CVIS. – 2018 – URL: https://www.researchgate.net/publication/ 330094226\_Action\_Recognition\_using\_Deep\_Convolutional\_Neural\_Networks\_and\_Compressed\_Spatio-Temporal\_Pose\_Encodings.
      33. Liu, M. Recognizing human actions as the evolution of pose estimation maps. / M.  Liu, J. Yuan. // CVPR. – 2018. – P.1159- 1168.
      34. McNally, W. Action Recognition using Spatio-Temporal Activation Reprojection. / W. McNally, A. Wong, J. McPhee. // STAR-Net in CVIS. – 2019. – P.4597-4605.
      35. Mayer, N. What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation? / N. Mayer, E. Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy, T. Brox. // International Journal of Computer Vision. – 2018. – 126(9) . – P.942–960.

Приложение

Основная часть программы:

function [output] = New\_optical\_flow\_4(imgs, Options)

t = tic;

for iframe = 1:1 % мы обрабатываем 2 последовательных кадра

display(sprintf('processing ...'))

%Считываем изображение

frame2\_1 = imread('turn\_1.png');

%Поворачиваем изображение (p(1) с чертой)

frame3\_1 = imrotate(frame2\_1,30,'bilinear','crop');

blk\_size\_y = size(frame2\_1,1);%кол-во пикселей по оси y

blk\_size\_x = size(frame2\_1,2);%кол-во пикселей по оси x

%------------------------------------------------------

%добавим шум в изображения

% frame2\_1=imnoise(frame2\_1,'gaussian',0.2);

% frame3\_1=imnoise(frame3\_1,'gaussian',0.2);

%------------------------------------------------------

x = zeros(1,blk\_size\_y\*blk\_size\_x);%заполняет нулями матрицу, указанной размерности

y = zeros(1,blk\_size\_y\*blk\_size\_x);

brightness\_2 = zeros(1,blk\_size\_y\*blk\_size\_x);

ct = 1;

for j = 1:blk\_size\_y

for i = 1:blk\_size\_x

%---------------------------------------------------------------------

% Задаём y(0), p(0), p(1) с чертой

x(ct) = i;%задаём координаты пикселей по строке

y(ct) = j;%задаём координаты пикселей по столбцу

brightness\_0(iframe,ct) = frame2\_1(j,i);%задаём значение яркости каждого пикселя-p(0)

brightness\_1(iframe,ct) = frame3\_1(j,i);% p(1) с чертой

ct = ct+1;

end

end

u0 = [0 0 0];%начальное приближение

u\_move = zeros(1,2);

u\_move(1) = u0(2);

u\_move(2) = u0(3);

u\_return = u0(1);

%Сначала определяем смещение (сдвиг)

u2\_3 = Grad\_descent(u\_move,x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

display(u2\_3);

%Сдвигаем изображение сообразно u2\_3

ct = 1;

for j = 1:blk\_size\_y

for i = 1:blk\_size\_x

%--------------------------------------------------

% 3. Вычисляем y(1) = y(0) + u(0)

x\_1(ct) = x(ct) + round(u2\_3(1));

y\_1(ct) = y(ct) + round(u2\_3(2));

%--------------------------------------------------

% 4. Вычисляем p(1,y(1))

% Если новые координаты находятся в пределах изображения, присваиваем им новые координаты яркости

if x\_1(ct)>= 1 & x\_1(ct) <= blk\_size\_x && y\_1(ct)>= 1 & y\_1(ct) <= blk\_size\_y

temp\_count = ct + round(u2\_3(1)) + blk\_size\_x \* round(u2\_3(2));

brightness\_0(iframe,temp\_count) = brightness\_0(iframe,ct);

end

ct = ct+1;

end

end

%Затем определяем поворот

[u1,g] = Grad\_descent\_2(u\_return,x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

display(u1);

display(g);

x\_1 = [];

y\_1 = [];

ct = 1;

u = zeros(1,3);

u(1) = u1;

u(2) = u2\_3(1);

u(3) = u2\_3(2);

%для корректировки смещения при повороте

x\_center = round(blk\_size\_x/2);

y\_center = round(blk\_size\_y/2);

x\_center\_rotation = x\_center\*cosd(u(1)) + y\_center\*sind(u(1));

y\_center\_rotation = -x\_center\*sind(u(1)) + y\_center\*cosd(u(1));

x\_move = x\_center\_rotation - x\_center;

y\_move = y\_center\_rotation - y\_center;

temp\_y = 2;

t = 1;

%Создаём массивы, с помощью которых будем отображать поле скоростей:

% pos - позиция вектора; u\_1,u\_2 - смещение по осям

for j = 1:blk\_size\_y

temp\_x = 1;

for i = 1:blk\_size\_x

x\_1(t) = round(x(ct)\*cosd(u(1)) + y(ct)\*sind(u(1)) - x\_move + u(2));

y\_1(t) = round(y(ct)\*cosd(u(1)) - x(ct)\*sind(u(1)) - y\_move + u(3));

u\_1(t) = x\_1(t)-x(ct);

u\_2(t) = y\_1(t)-y(ct);

pos(1,t) = temp\_x;

pos(2,t) = temp\_y;

ct = ct + 1;

t = t + 1;

temp\_x = temp\_x + 1;

end

temp\_y = temp\_y + 1;

end

ct = 1;

t = 1;

%Создаём массивы, с помощью которых будем отображать "прореженное" поле скоростей

for j = 1:19

for i = 1:15

u\_1\_1(t) = u\_1(ct);

u\_2\_1(t) = u\_2(ct);

pos\_1(1,t) = pos(1,ct);

pos\_1(2,t) = pos(2,ct);

ct = ct + 5;

t = t + 1;

end

ct = ct + 225;

end

end

%Отображаем поле скоростей

figure

subplot(1,1,1)

frame\_clean = zeros(blk\_size\_y,blk\_size\_x);

frame\_clean = 1;

imagesc(frame\_clean),colormap bone

axis image

hold on

xx = squeeze(pos\_1(2,:));

yy = squeeze(pos\_1(1,:));

vx = squeeze(u\_2\_1(:));

vy = squeeze(u\_1\_1(:));

display(size(xx));

display(size(yy));

display(size(vx));

display(size(vy));

plot(yy,xx,'.k','markersize',1)

quiver(yy,xx,vy',vx','r','linewidth',1.3), hold off

figure

subplot(1,2,1)

imagesc(frame2\_1),colormap bone

axis image

subplot(1,2,2)

imagesc(frame3\_1),colormap bone

axis image

%------------------------------------------------------------------------

% Использовать этот кусок кода, если необходимо разбить изображение на отдельно рассматриваемые части

% frame2\_1\_1 = frame2\_1;

% frame3\_1\_1 = frame3\_1;

% frame2\_1 = [];

% frame3\_1 = [];

% [blk\_size\_yy, blk\_size\_xx, z] = size(frame2\_1\_1);

% display(blk\_size\_yy);

% display(blk\_size\_xx);

% display(z);

% blk\_size\_y = floor(blk\_size\_yy\*0.33);

% blk\_size\_x = floor(blk\_size\_xx\*1);

%

% u = zeros(3,floor(blk\_size\_yy/blk\_size\_y)\*floor(blk\_size\_xx/blk\_size\_x));

% pos = zeros(2,floor(blk\_size\_yy/blk\_size\_y)\*floor(blk\_size\_xx/blk\_size\_x));

% t = 1;

% step\_y\_1 = 1;

% for n = 1:floor(blk\_size\_yy/blk\_size\_y)

% step\_y\_2 = step\_y\_1 + blk\_size\_y - 1;

% step\_x\_1 = 1;

% for m = 1:floor(blk\_size\_xx/blk\_size\_x)

% step\_x\_2 = step\_x\_1 + blk\_size\_x - 1;

%

% frame2\_1 = frame2\_1\_1(step\_y\_1:step\_y\_2,step\_x\_1:step\_x\_2);

% frame3\_1 = frame3\_1\_1(step\_y\_1:step\_y\_2,step\_x\_1:step\_x\_2);

%

% x = zeros(1,blk\_size\_y\*blk\_size\_x);

% y = zeros(1,blk\_size\_y\*blk\_size\_x);

% x\_1 = zeros(1,blk\_size\_y\*blk\_size\_x);

% y\_1 = zeros(1,blk\_size\_y\*blk\_size\_x);

% x\_2 = zeros(1,blk\_size\_y\*blk\_size\_x);

% y\_2 = zeros(1,blk\_size\_y\*blk\_size\_x);

% brightness\_2 = zeros(1,blk\_size\_y\*blk\_size\_x);

%

% ct = 1;

% for j = 1:blk\_size\_y

% for i = 1:blk\_size\_x

%

% %---------------------------------------------------------------------

% % Задаём y(0), p(0), p(1) с чертой

% x(ct) = i;%задаём координаты пикселей по строке

% y(ct) = j;%задаём координаты пикселей по столбцу

% brightness\_0(iframe,ct) = frame2\_1(j,i);%задаём значение яркости каждого пикселя-p(0)

% brightness\_1(iframe,ct) = frame3\_1(j,i);% p(1) с чертой

%

% ct = ct+1;

% end

% end

%

% %Сначала определяем смещение (сдвиг)

% u2\_3 = Grad\_descent(u\_move,x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

% display(u2\_3);

% %Затем определяем поворот

% [u1,g] = Grad\_descent\_2(u\_return,x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

% display(u1);

% display(g);

%

% x\_1 = [];

% y\_1 = [];

% ct = 1;

% u = zeros(1,3);

% u(1) = u1;

% u(2) = u2\_3(1);

% u(3) = u2\_3(2);

% u\_all(:,t) = u;

%

% t = t+1;

% step\_x\_1 = step\_x\_2+1;

% end

% step\_y\_1 = step\_y\_2+1;

% end

% %Задаём поле скоростей для каждого рассматриваемого блока

% for m = 1:floor(blk\_size\_yy/blk\_size\_y)\*(blk\_size\_xx/blk\_size\_x)

% %для корректировки смещения при повороте

% x\_center = round(blk\_size\_x/2);

% y\_center = round(blk\_size\_y/2);

% x\_center\_rotation = x\_center\*cosd(u(1)) + y\_center\*sind(u(1));

% y\_center\_rotation = -x\_center\*sind(u(1)) + y\_center\*cosd(u(1));

% x\_move = x\_center\_rotation - x\_center;

% y\_move = y\_center\_rotation - y\_center;

% temp\_y = 2;

% t = 1;

% %Создаём массивы, с помощью которых будем отображать поле скоростей:

% % pos - позиция вектора; u\_1,u\_2 - смещение по осям

% for j = 1:blk\_size\_y

% temp\_x = 1;

% for i = 1:blk\_size\_x

% x\_1(t) = round(x(ct)\*cosd(u(1)) + y(ct)\*sind(u(1)) - x\_move + u(2));

% y\_1(t) = round(y(ct)\*cosd(u(1)) - x(ct)\*sind(u(1)) - y\_move + u(3));

% u\_1(t) = x\_1(t)-x(ct);

% u\_2(t) = y\_1(t)-y(ct);

% pos(1,t) = temp\_x;

% pos(2,t) = temp\_y;

% ct = ct + 1;

% t = t + 1;

% temp\_x = temp\_x + 1;

% end

% temp\_y = temp\_y + 1;

% end

% ct = 1;

% t = 1;

% %Создаём массивы, с помощью которых будем отображать "прореженное" поле скоростей

% for j = 1:19

% for i = 1:15

% u\_1\_1(t) = u\_1(ct);

% u\_2\_1(t) = u\_2(ct);

% pos\_1(1,t) = pos(1,ct);

% pos\_1(2,t) = pos(2,ct);

% ct = ct + 5;

% t = t + 1;

% end

% ct = ct + 225;

% end

% %Отображаем поле скоростей

% figure

% subplot(1,1,1)

% frame\_clean = zeros(blk\_size\_y,blk\_size\_x);

% frame\_clean = 1;

% imagesc(frame\_clean),colormap bone

% axis image

% hold on

% xx = squeeze(pos\_1(2,:));

% yy = squeeze(pos\_1(1,:));

% vx = squeeze(u\_2\_1(:));

% vy = squeeze(u\_1\_1(:));

% display(size(xx));

% display(size(yy));

% display(size(vx));

% display(size(vy));

% plot(yy,xx,'.k','markersize',1)

% quiver(yy,xx,vy',vx','r','linewidth',1.3), hold off

%

% figure

% subplot(1,2,1)

% imagesc(frame2\_1),colormap bone

% axis image

% subplot(1,2,2)

% imagesc(frame3\_1),colormap bone

% axis image

% end

Функция Grad\_descent:

function u = Grad\_descent(u,x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

alpha = 1 ;

G = Grad\_Val(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

g = G(u);

k = 0;

f1 = 0;

f2 = 0;

while norm(g) > 0.01%1e-2

d = - g ;

f1 = f2;

alpha = Length\_of\_Step(u,g,d,x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

u = u + alpha \* d;

display(g) ;

display(alpha) ;

if k > 100

display('Too many iterations');

break;end

k = k + 1 ;

p = make\_temp\_count\_M(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

f2 = p(u);

display(u) ;

display(f2) ;

if abs(f1 - f2) < 0.1

display('|f(u)-f(u+1)| < 1');

break;end

G = Grad\_Val(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

g = G(u);

end

%disp(u) ;

end

Функция Grad\_Val:

function G = Grad\_Val(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

G = @(u) grad\_val (u);

function g = grad\_val(u)

ct = 1;

for j = 1:blk\_size\_y %xs % движемся по строке

for i = 1:blk\_size\_x

%--------------------------------------------------

% 3. Вычисляем y(1) = y(0) + u(0)

x\_1(ct) = x(ct) + round(u(1));

y\_1(ct) = y(ct) + round(u(2));

%--------------------------------------------------

% 4. Вычисляем p(1,y(1))

% Если новые координаты находятся в пределах изображения, присваеваем им новые координаты яркости

if x\_1(ct)>= 1 & x\_1(ct) <= blk\_size\_x && y\_1(ct)>= 1 & y\_1(ct) <= blk\_size\_y

temp\_count = ct + round(u(1)) + blk\_size\_x \* round(u(2));

brightness\_2(iframe,temp\_count) = brightness\_0(iframe,ct);

end

ct = ct+1;

end

end

%---------------------------------------------------------------------

%5. Начало расчета dp/dy

dp\_dy = zeros(blk\_size\_y\*blk\_size\_x,2);

% Dp\_Dy1 = zeros(blk\_size\_y,blk\_size\_x);

% Dp\_Dy2 = zeros(blk\_size\_y,blk\_size\_x);

ct = blk\_size\_x+2;

%Центральные пиксели

for j = 1:blk\_size\_y-2

for i = 1:blk\_size\_x-2 % движемся по строке

dp\_dy(ct,1) = (-1)\*brightness\_2(iframe,ct-blk\_size\_x-1) + (-2)\*brightness\_2(iframe,ct-1) + (-1)\*brightness\_2(iframe,ct+blk\_size\_x-1) + brightness\_2(iframe,ct-blk\_size\_x+1) + 2\*brightness\_2(iframe,ct+1) + brightness\_2(iframe,ct+blk\_size\_x+1);

dp\_dy(ct,2) = (-1)\*brightness\_2(iframe,ct-blk\_size\_x-1) + (-2)\*brightness\_2(iframe,ct-blk\_size\_x) + (-1)\*brightness\_2(iframe,ct-blk\_size\_x+1) + brightness\_2(iframe,ct+blk\_size\_x-1) + 2\*brightness\_2(iframe,ct+blk\_size\_x) + brightness\_2(iframe,ct+blk\_size\_x+1);

ct = ct + 1;

end

ct = ct + 2;

end

ct = 1;

%Конец расчета dp/dy

%------------------------------------------------------

% 7. Gradient [dp\_dy1;dp\_dy2]

g = NaN;

g(1) = -2\*(double(brightness\_2) - double(brightness\_1))\*dp\_dy(:,1);

g(2) = -2\*(double(brightness\_2) - double(brightness\_1))\*dp\_dy(:,2);

brightness\_2 = zeros(1,blk\_size\_y\*blk\_size\_x);

end

end

Функция Length\_of\_Step:

function alpha = Length\_of\_Step(u,g,d,x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

alpha = 1.0;

k = 1;

beta = 0.1 ;

tau = 0.5 ;

p = make\_temp\_count\_M(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

f = p(u);

display(size(g));

while p(u + alpha\*d) >= (f + alpha\* beta\*(g \* d'))

alpha = tau \* alpha ;

if k > 100

break;end

k = k +1;

end

end

Функция make\_temp\_count\_M:

function p = make\_temp\_count\_M(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

p = @(u) f\_val\_grad\_val (u);

function I = f\_val\_grad\_val(u)

ct = 1;

for j = 1:blk\_size\_y

for i = 1:blk\_size\_x

%--------------------------------------------------

% 3. Вычисляем y(1) = y(0) + u(0)

x\_1(ct) = x(ct) + round(u(1));

y\_1(ct) = y(ct) + round(u(2));

%--------------------------------------------------

% 4. Вычисляем p(1,y(1))

% Если новые координаты находятся в пределах изображения, присваеваем им новые координаты яркости

if x\_1(ct)>= 1 & x\_1(ct) <= blk\_size\_x && y\_1(ct)>= 1 & y\_1(ct) <= blk\_size\_y

temp\_count = ct + round(u(1)) + blk\_size\_x \* round(u(2));

brightness\_2(iframe,temp\_count) = brightness\_0(iframe,ct);

end

ct = ct+1;

end

end

% 6. Вычисляем p(1,y(1))

I = sum((double(brightness\_2) - double(brightness\_1)).^2);

brightness\_2 = zeros(1,blk\_size\_y\*blk\_size\_x);

end

end

Функция Grad\_Val:

function G = Grad\_Val(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

G = @(u) grad\_val (u);

function g = grad\_val(u)

ct = 1;

for j = 1:blk\_size\_y

for i = 1:blk\_size\_x

%--------------------------------------------------

% 3. Вычисляем y(1) = y(0) + u(0)

x\_1(ct) = x(ct) + round(u(1));

y\_1(ct) = y(ct) + round(u(2));

%--------------------------------------------------

% 4. Вычисляем p(1,y(1))

% Если новые координаты находятся в пределах изображения, присваеваем им новые координаты яркости

if x\_1(ct)>= 1 & x\_1(ct) <= blk\_size\_x && y\_1(ct)>= 1 & y\_1(ct) <= blk\_size\_y

temp\_count = ct + round(u(1)) + blk\_size\_x \* round(u(2));

brightness\_2(iframe,temp\_count) = brightness\_0(iframe,ct);

end

ct = ct+1;

end

end

%------------------------------------------------------

%5. Начало расчета dp/dy

dp\_dy = zeros(blk\_size\_y\*blk\_size\_x,2);

ct = blk\_size\_x+2;

%Центральные пиксели

for j = 1:blk\_size\_y-2

for i = 1:blk\_size\_x-2 % движемся по строке

dp\_dy(ct,1) = (-1)\*brightness\_2(iframe,ct-blk\_size\_x-1) + (-2)\*brightness\_2(iframe,ct-1) + (-1)\*brightness\_2(iframe,ct+blk\_size\_x-1) + brightness\_2(iframe,ct-blk\_size\_x+1) + 2\*brightness\_2(iframe,ct+1) + brightness\_2(iframe,ct+blk\_size\_x+1);

dp\_dy(ct,2) = (-1)\*brightness\_2(iframe,ct-blk\_size\_x-1) + (-2)\*brightness\_2(iframe,ct-blk\_size\_x) + (-1)\*brightness\_2(iframe,ct-blk\_size\_x+1) + brightness\_2(iframe,ct+blk\_size\_x-1) + 2\*brightness\_2(iframe,ct+blk\_size\_x) + brightness\_2(iframe,ct+blk\_size\_x+1);

ct = ct + 1;

end

ct = ct + 2;

end

ct = 1;

%Конец расчета dp/dy

%------------------------------------------------------

% 7. Gradient [dp\_dy1;dp\_dy2]

g = NaN;

g(1) = -2\*(double(brightness\_2) - double(brightness\_1))\*dp\_dy(:,1);

g(2) = -2\*(double(brightness\_2) - double(brightness\_1))\*dp\_dy(:,2);

brightness\_2 = zeros(1,blk\_size\_y\*blk\_size\_x);

end

end

Функция Grad\_descent\_2:

function [u,g] = Grad\_descent\_2(u,x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

alpha = 1 ;

G = Grad\_Value(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

g = G(u);

k = 0;

f1 = 0;

f2 = 0;

p = F\_Val(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

f3 = p(u);% начальное приближение

while norm(g) > 1e-2

d = - g ;

f1 = f2;

alpha = Length\_of\_Step\_2(u,g,d,x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

u = u + alpha \* d;

if k > 100

display('Too many iterations');

break;end

k = k + 1 ;

f2 = p(u);

if (f3 - f2) < 0

display('f(u)-f(u+1) < 0');

break;end

f3 = f2;

if abs(f1 - f2) < 0.1

display('|f(u)-f(u+1)| < 0.1');

break;end

G = Grad\_Value(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

g = G(u);

end

disp(u-360\*(floor(u/360)));

end

Функция Grad\_Value:

function G = Grad\_Value(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

G = @(u) grad\_val (u);

function g = grad\_val(u)

%---------Вычисляем brightness\_2----------------------

ct = 1;

for j = 1:blk\_size\_y

for i = 1:blk\_size\_x

B0(j,i) = brightness\_0(1,ct);

ct = ct + 1;

end

end

frame\_new = imrotate(B0,u,'bilinear','crop');

ct=1;

for j = 1:blk\_size\_y

for i = 1:blk\_size\_x

brightness\_2(iframe,ct) = frame\_new(j,i);

ct = ct+1;

end

end

%---------------------------------------------------------------

%5. Начало расчета dp/dy

dp\_dy = zeros(blk\_size\_y\*blk\_size\_x,2);

ct = blk\_size\_x+2;

%Центральные пиксели

for j = 1:blk\_size\_y-2

for i = 1:blk\_size\_x-2

dp\_dy(ct,1) = (-1)\*brightness\_2(iframe,ct-blk\_size\_x-1) + (-2)\*brightness\_2(iframe,ct-1) + (-1)\*brightness\_2(iframe,ct+blk\_size\_x-1) + brightness\_2(iframe,ct-blk\_size\_x+1) + 2\*brightness\_2(iframe,ct+1) + brightness\_2(iframe,ct+blk\_size\_x+1);

dp\_dy(ct,2) = (-1)\*brightness\_2(iframe,ct-blk\_size\_x-1) + (-2)\*brightness\_2(iframe,ct-blk\_size\_x) + (-1)\*brightness\_2(iframe,ct-blk\_size\_x+1) + brightness\_2(iframe,ct+blk\_size\_x-1) + 2\*brightness\_2(iframe,ct+blk\_size\_x) + brightness\_2(iframe,ct+blk\_size\_x+1);

ct = ct + 1;

end

ct = ct + 2;

end

%Конец расчета dp/dy

%------------------------------------------------------

% 7. Gradient

g = NaN;

ct = 1;

a = zeros(1,blk\_size\_y\*blk\_size\_x);

back = int16(blk\_size\_y\*blk\_size\_x);

for j = 1:blk\_size\_y

for i = 1:blk\_size\_x

p1(ct) = 2\*(double(brightness\_2(1,ct)) - double(brightness\_1(1,ct)))\*dp\_dy(ct,1);

p2(ct) = 2\*(double(brightness\_2(1,ct)) - double(brightness\_1(1,ct)))\*dp\_dy(ct,2);

a(ct) = p1(ct)\*(y(ct)\*cosd(u(1)) - x(ct)\*sind(u(1)))-p2(ct)\*(x(ct)\*cosd(u(1)) + y(ct)\*sind(u(1)));

back = back - 1;

ct = ct + 1;

end

end

g(1) = sum(a);

brightness\_2 = zeros(1,blk\_size\_y\*blk\_size\_x);

end

end

Функция F\_Val:

function p = F\_Val(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

p = @(u) f\_val (u);

function I = f\_val(u)

%---------Вычисляем brightness\_2---------------------------

ct = 1;

for j = 1:blk\_size\_y %xs % движемся по строке

for i = 1:blk\_size\_x

B0(j,i) = brightness\_0(1,ct);

ct = ct + 1;

end

end

frame\_new = imrotate(B0,u,'bilinear','crop');

ct=1;

for j = 1:blk\_size\_y

for i = 1:blk\_size\_x

brightness\_2(iframe,ct) = frame\_new(j,i);%задаём значение яркости каждого пикселя-p(0)

ct = ct+1;

end

end

%----------------------------------------------------------

% 6. Вычисляем p(1,y(1))

I = sum((double(brightness\_2) - double(brightness\_1)).^2);

brightness\_2 = zeros(1,blk\_size\_y\*blk\_size\_x);

end

end

Функция Length\_of\_Step\_2:

function alpha = Length\_of\_Step\_2(u,g,d,x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe)

alpha = 1.0;

k = 1;

beta = 0.1 ;

tau = 0.5 ;

p = F\_Val(x,y,x\_1,y\_1,blk\_size\_x,blk\_size\_y,brightness\_0,brightness\_1,brightness\_2,iframe);

f = p(u);

f\_alpha\_2 = f;

while p(u + alpha\*d) >= f;%(f + alpha\*beta\*(g\*(d')))

alpha = tau \* alpha ;

if k > 100

break;end

k = k +1;

end

end