Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Реализация субпиксельного уточнения ViFlow метода поиска оптического потока.

Курсовая работа студента 445 группы
Расторгуева Алексея Сергеевича

Научный руководитель …….…. Пименов А.

Санкт-Петербург
2010

Оглавление

Введение. Постановка задачи

Сегодня оптические потоки все чаще становятся предметом университетских лекций и темой международных конференций. Само словосочетание "оптический поток" приобретает устойчивость и выходит за рамки узкоспециализированных научных диалектов: сейчас поисковый сервер google выдает в ответ на запрос "optical flow" 8 030 000 результатов.

Каждый источник, предоставляющий информацию об оптических потоках, стремится дать свое определение этому объекту. Вот некоторые из них:

- Оптический поток это рисунок видимого движения объектов, поверхностей или краев сцены, получаемый в результате перемещения наблюдателя (глаз или камеры) относительно сцены (Википедия).
- Оптический поток – движение отдельных пикселей видеоряда [1].

Оптические потоки все более расширяют границу их применения: они интенсивно используются для создания видео спецэффектов, сжатии видео, машинном зрении и многих других областях. В оптическом потоке зашифрована полезная информация о структуре сцены. Например, при наблюдении из движущегося автомобиля удаленные объекты характеризуются гораздо более медленным кажущимся движением по сравнению с близкими объектами, поэтому скорость кажущегося движения позволяет получить определенную информацию о расстоянии. Косвенным подтверждением важности оптических потоков может также служить внушительный арсенал алгоритмических средств, применяемых для их получения: используются, в частности, цифровая фильтрация, Фурье- и wavelet-преобразования, моделирование нейронных сетей и всевозможные разновидности метода конечных элементов. В некоторых приложениях, требующих быстрой реакции (например, в системах машинного зрения), для извлечения оптических потоков "в реальном времени" используются специализированные сигнальные процессоры и параллельные вычисления.

В данной работе делается попытка получения алгоритма для повышения точности вычисления метода поиска оптического потока ViFlow, а также реализация и сравнение с этим методом по некоторым параметрам другого метода поиска оптического потока – KLT алгоритма, для дальнейшего применения их в некоторых прикладных задачах, в

частности, в задаче стереозрения и задаче повышения разрешения видеопотока. (superresolution).

Для многих алгоритмов, применяющих оптические потоки, точность найденного потока должна быть выше, чем один пиксель. Так KLT алгоритм обладает субпиксельной точностью, в отличие от ViFlow.

В задаче стереозрения использование таких алгоритмов может в несколько раз повысить скорость работы. В общем, стереозрение включает два процесса: совмещение деталей, наблюдаемых двумя и (или большим числом камерами) и восстановление их трехмерного прообраза. Первый процесс можно свести к поиску для каждой конкретной точки изображения полученного с первой камеры соответствующей точки на некоторой прямой на изображении со второй камеры. Использование субпиксельных методов поиска потока на этом шагу может повысить скорость поиска в 2-3 раза. Последний процесс относительно прост: прообраз соответствующих точек можно найти как точку пересечения лучей, приходящих через эти точки и центры соответствующих диафрагм камер[1].

Основной целью работы было субпиксельное уточнение существующей реализации метода ViFlow, а также получение реализации метода KLT, для сравнения результатов работы и возможного использования в дальнейшем.

Предлагаемое решение

Субпиксельное уточнение метода ViFlow было решено делать на основе какого-либо из уже существующих субпиксельных методов поиска потока, т.е. применение субпиксельного алгоритма к результату работы ViFlow.

Многие современные алгоритмы поиска потока основаны на идее Лукаса и Канаде[2]. Так

1. Lucas-Kanade – точка считается только смещающейся, без искажений.
2. Tomassi-Kanade – переформулирование Lucas-Kanade. Движение считается смещение, и рассчитывается путем итеративного решения построенной системы линейных уравнений
3. Shi-Tomassi-Kanade – учитываются аффинные искажения окрестности точки.
4. Jin-Favaro-Soatto – модификация Shi-Tomassi-Kanade с учетом аффинных изменений освещенности окрестности точки.

Метод ViFlow сопоставляет точки двух изображений, сравнивая хэш-функции от их окрестностей. Точка первого изображения переходит в точку второго изображения, если эти значения равны. В этом случае окрестности точек сильно похожи, и для уточнения достаточно применять только метод Лукаса-Канаде.

Т.к. на данном этапе хочется искать поток в видеопоследовательности, в которой последовательные кадры сильно похожи, нет необходимости использовать эти методы. Для соседних кадров изменения незначительны и можно считать, что ни окрестность точки, ни освещение не искажаются, поэтому и был выбран для сравнения только первый метод.

Описание существующих алгоритмов

Как уже было сказано, одним из основных алгоритмов, рассмотренных в данной работе, является алгоритм Канаде - Лукаса. Алгоритм, описанный в [3] состоит из двух шагов:

1)Выбор особых точек.

2)Отслеживание особых точек.

Общая цель алгоритма – для точки u поиск вектора d, минимизирующего следующую сумму

$$\sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (I(x,y) - J(x+d_x, y+d_y))^2$$

, где I(x,y),J(x,y) – яркости первого и второго изображений в точке (x,y).

Имеется несколько вариаций алгоритма Канаде - Лукаса. Два из них – классический и пирамидальный алгоритмы. Рассмотрим классический метод.

Процесс вычисления протекает итеративно, используя метод Ньютона, и имеет следующие шаги

1) Вычисление матрицы пространственного градиента G, имеющей следующий вид

$$\iint_{W(x_0,y_0)} \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial x \partial y} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix} dxdy$$

где $W(x_0,y_0)$ – прямоугольная окрестность точки , в которой ищется поток.

2) Вычисление вектора ошибки e

$$\mathbf{e} = \int\int_W [I(\mathbf{x}) - J(\mathbf{x})]\mathbf{g}(\mathbf{x})\, d\mathbf{x}$$

3) Вычисление вектора уточнения d из уравнения G * d = e

4) Сложения текущего вектора потока с уточнением.

В качестве начального приближения берется нулевой вектор.

Пирамидальный (иерархический) KLT алгоритм в значительной степени использует классический метод. В методе используется пирамида изображений т.е. последовательность изображений, в которой каждое последующее получается из предыдущего уменьшением его размеров в два раза. Каждое изображение последовательности, за исключением первого, получается как свертка предыдущего изображения со следующим фильтром: $[¼\ \ ½\ ¼] * [¼\ \ ½\ ¼]^T$ т.е.

$$I^L(x,y) = 1/4 * (I^{L-1}(2x,2y) + 1/8 * (I^{L-1}(2x-1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y+1)) + 1/16 * (I^{L-1}(2x-1,2y-1) + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y-1) + I^{L-1}(2x+1,2y+1))$$

Пирамидальный KLT алгоритм для поиска потока в точке (x_0,y_0):

1)По каждому из двух данных изображениям строится пирамида изображений.

2)Для i – го изображения пирамиды по первому и второму изображениям применяется классический KLT метод, с вектором начального приближения $2 * d_{i+1}$, где d_{i+1} – вектор потока, полученный на предыдущем уровне пирамиды. Для самого первого уровня этот вектор принимается равным (0,0).

Вторым алгоритмом, рассмотренным в данной работе, является алгоритм ViFlow. Алгоритм заключается в следующем:

1) Для каждой точки первого и второго изображений вычисляется hash - функция от окрестности этой точки. Таким образом, для каждого изображения получается массив значений этой функции.

2) Каждый из полученных массивов сортируется.

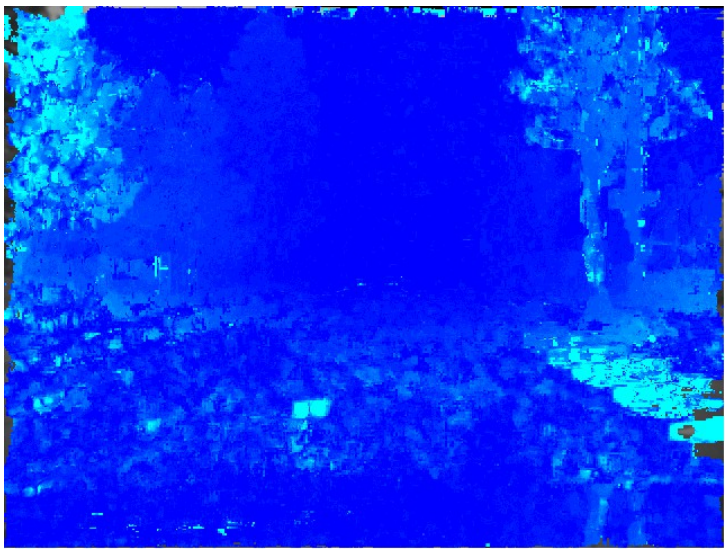
3) Считается, что точка u первого изображения переходит в точку v второго изображения, если соответствующие им значения хэша равны

Особенности реализации

Основные структуры, используемые в программе – буфер изображения G12Buffer, хранящий информацию о длине и ширине изображения и указатель на область данных типа double и буфер потока также хранящий информацию о длине и ширине и указатель на данные типа вектора с целыми координатами vector2Ds16.
В программе функция
calculateHierarchicalKLTFlow(G12Buffer *buffer1,G12Buffer *buffer2,KLTHierarchicalContext *context).
вычисляет поток методом Канаде-Лукаса. Функция принимает на вход 2 изображения и контекст и возвращает указатель на участок памяти, содержащий буфер потока.

Контекст представляет структуру хранящую информацию о размерах окна интегрирования, количестве итераций метода Ньютона, количестве уровней пирамиды изображений, структуре низкочастотного фильтра и коэффициенты, используемые для фильтрации потока во время поиска.

Как видно из рис.1 (поток для изображений 2 и 3) вблизи границы и центра изображений, где яркость пикселей меняется мало, поток вычисляется достаточно неточно. На рис.1 в центральной области видны 2 пятна – движение камней на асфальте, в остальном окружающей поток имеет меньшую величину из-за похожей яркости точек асфальта. Это неверно, так как окружающие точки движутся примерно с такой же скоростью. Дело в том, что метод Лукаса - Канаде дает хороший результат для точек, имеющих достаточно большие собственные числа матрицы G(или также определитель матрицы пространственного градиента). В основном, это точки находящиеся вблизи границ различных объектов, чья яркость заметно отличается от яркости окружающей среды. Для удаления маловероятного потока производилась его фильтрация во время работы, основываясь на величине определителя матрицы пространственного градиента.



(рис.1) Изображение потока при высоте пирамиды 2,4 итерациях метода Ньютона.

Фильтрация происходила следующим образом: если на данном уровне пирамиды определитель обратной матрицы G меньше заранее определенного порога, вектор потока считается неизвестным. Пороги для каждого уровня определяются опытным путем. Было определено, что хорошие результаты для данного формата изображений (используются 12ти битные изображения) дают величины порядка 1-10 * 1000 * 2^{24} для каждого уровня пирамиды (рис. 5,6).

Фильтрация потока на нижних уровнях, в основном используется для ускорения вычислений. Если в некоторой точке u на уровне L поток не определен, в каждой точке следующего уровня, которой соответствует эта точка, считается, что поток также не определен и дальнейшие вычисления для этой точки не производятся, что для представленных коэффициентов заметно снижает время вычисления (табл.1,2). При увеличении коэффициентов на нижних уровнях повышается скорость работы программы, но уменьшается плотность выходного потока.

В работе Канаде и Лукаса для поиска “хороших” для отслеживания точек предлагается следующий алгоритм: среди всех точек ищется максимальное значение max наибольшего модуля собственного числа матрицы G.Далее выбираются точки у которых максимальное значение модуля собственного числа больше чем filteringPercent * max, где filteringPercent задает нужный процент от максимальной величины[3].

Фильтрация, использующаяся в данной работе по сути то же самое, также отбрасывает точки, величина большего собственного числа или определителя матрицы

пространственного градиента меньше некоторого порога, но позволяет избавиться от предобработки изображения, и, следовательно работает немного быстрее. На рис.6 изображены точки, полученные методом предложенным Канаде и Лукасом при различных значениях filteringPercent. На рис.4 справа видно, что поток для заданных коэффициентов вычисляется в тех же точках.

Для уменьшения объема вычислений при нахождении матрицы пространственных градиентов G был введен буфер интеграла пространственного градиента, представляющий структуру, хранящую информацию о длине и ширине изображения и интегралы по прямоугольной области от начала координат до заданной точки от квадратов производных интенсивности по x и y и произведения производных интенсивности по x и y. Используя определения

SIx2(intBuf, x, y) (intBuf)->data[3*y*(intBuf)->w+3*x]
SIxIy(intBuf, x,y) (intBuf)->data[3*y*(intBuf)->w+3*x+1]
SIy2(intBuf,x,y) (intBuf)->data[3*y*(intBuf)->w+3*x+2]

, где intBuf – интегральный буфер, элементы матрицы пространственного градиента по окрестности $[x_0-w_1, x_0+w_2] * [y_0-w_3, y_0+w_4]$ вычисляется как
g_{11} = SIx2(intBuf, x_0+w_2, y_0+w_4) + SIx2(intBuf, x_0+w_1, y_0+w_3) - SIx2(intBuf, x_0+w_2, y_0+w_3) - SIx2(intBuf, x_0+w_1, y_0+w_4)
g_{12} = SIxIy(intBuf, x_0+w_2, y_0+w_4) + SIxIy(intBuf, x_0+w_1, y_0+w_3) - SIxIy(intBuf, x_0+w_2, y_0+w_3) - SIxIy(intBuf, x_0+w_1, y_0+w_4)
$g_{21} = g_{12}$
g_{22} = SIy2(intBuf, x_0+w_2, y_0+w_4) + SIy2(intBuf, x_0+w_1, y_0+w_3) - SIy2(intBuf, x_0+w_2, y_0+w_3) - SIy2(intBuf, x_0+w_1, y_0+w_4)

Применение данной структуры позволяет в несколько раз снизить объем вычислений матрицы пространственного градиента и повысить общею скорость выполнения на величину порядка 4-6%. Также была введена более общая структура, в которой хранятся соответствующие буферы для каждого уровня пирамиды. Перед применением самого пирамидального KLT к изображениям, сначала строится пирамида интегральных буферов и далее вычисления матрицы пространственного градиента для каждого уровня происходит, используя эту пирамиду.

В отличие от алгоритма, описанного в работе Канаде – Лукаса, при вычислении потока для всех точек изображения были поменяны местами цикла по уровням пирамиды и точкам изображения, что позволило увеличить скорость работы более чем в 3 раза.
Данный алгоритм достаточно быстр, но, как видно на рис.7 имеет большой разброс. Для уменьшения разброса потока используется следующая фильтрация: если из окрестности данной точки выходит вектор в окрестности N различных точек, поток в данной точке объявляется неизвестным. Результат фильтрации приведен на рис 7. В отличие от фильтрации, применяемой в KLT методе, фильтрация ViFlow применяется к уже вычисленному потоку и увеличивает объем вычислений на величину порядка 4%.
Еще одним недостатком данного алгоритма является то, что он не имеет субпиксельной точности.

Для повышения точности вычисления, к потоку, полученному методом ViFlow, применялся классический KLT метод: просматривается полученный буфер потока и к каждой точке изображения, в которой известен поток применяется метод Ньютона который используется в классическом KLT алгоритме, причем за начальное приближение берется вектор, полученный методом ViFlow.

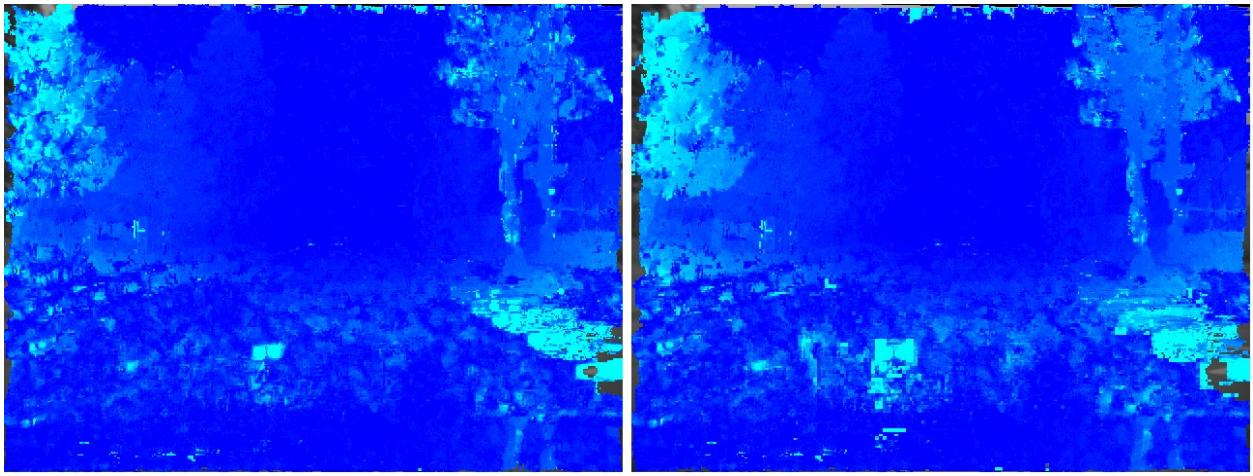
Таким образом, уточненный ViFlow метод представляет следующее:
1)Для двух изображений применяется обычный ViFlow.
2)Полученный поток фильтруется.
3)Для каждой точки изображения, поток в которой определен, применяется функция flowRectification, выдающая вектор уточнения.
Функция flowRectificationForPixel(G12Buffer *buffer1,vector2Ds16 point1,G12Buffer *buffer2,vector2Ds16 point2,KLTClassicalContext context) вычисляет уточнение потока, используя классический KLT алгоритм, принимает на вход указатели на оба буфера изображений buffer1 и buffer2, точку первого изображения point1, для которой производится уточнение потока, точку point2 второго изображения, для которой выполняются условия
point2.x = point1.x + flow.x
point2.y = point1.y + flow.y
, где flow – вектор потока в точке point1.
При этом в ходе вычисления также производится фильтрация.

Результаты

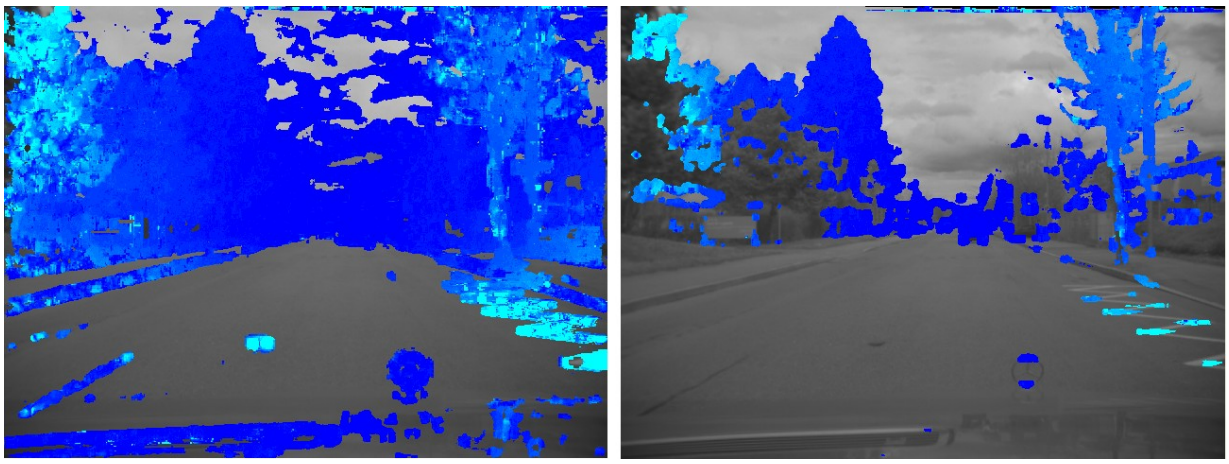
Ниже приводятся примеры работы методов KLT и ViFlow. На изображения 4,5,8 поток изображен не в виде направления, а как модуль величины потока в данной точке, т.е. интенсивность голубого цвета в каждой точке в которой поток определен тем выше, чем больше модуль потока в данной точке.

(рис.2)

(рис.3)

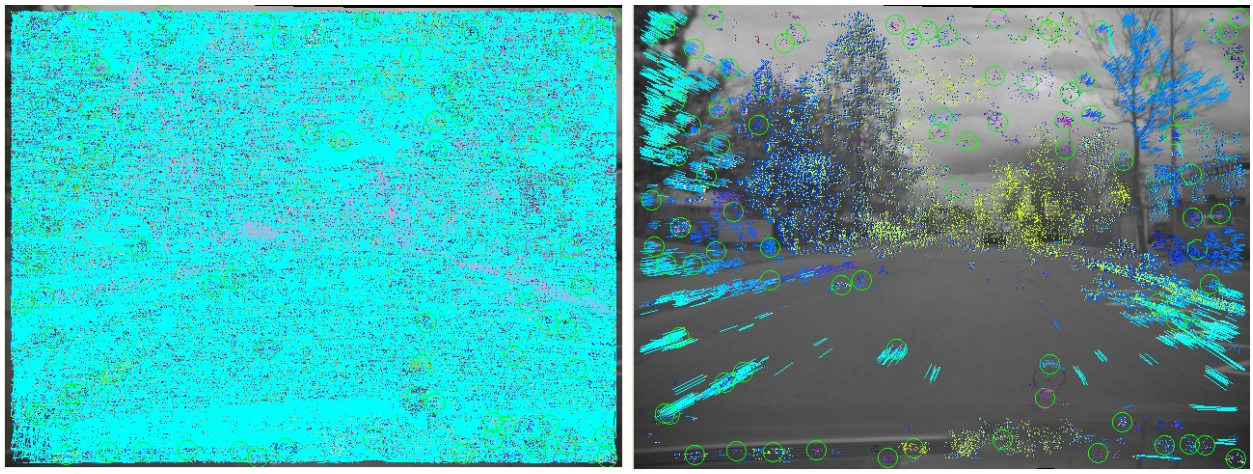


(рис. 4)Изображения нефильтрованного потока между рис.2 и рис.3.Слева высота пирамиды 2,итераций метода Ньютона 4.Справа высота 3, 4 итерации метода Ньютона.

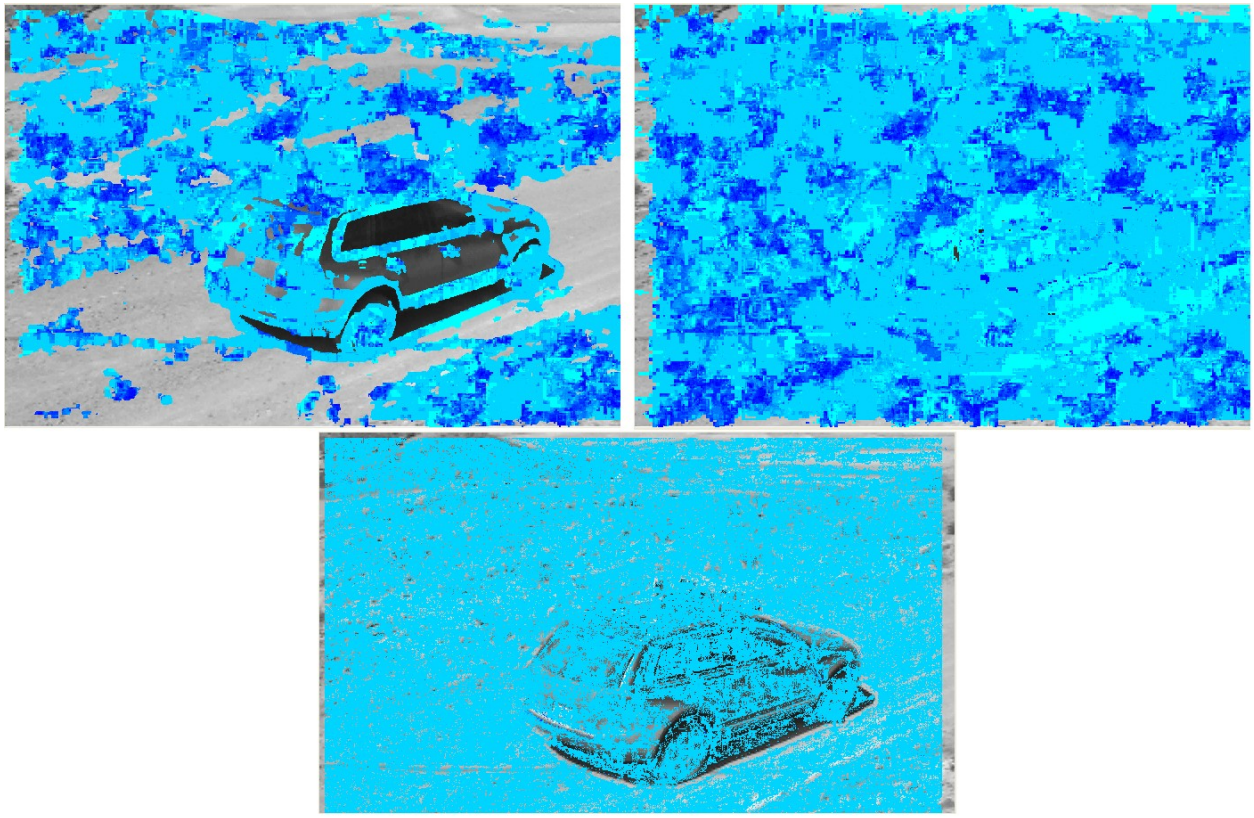


(рис.5)Изображение фильтрованного потока, сравнением на каждой итерации по высоте пирамиды с порогом. Высота 3, 4 итерации. Слева коэффициенты фильтрации $4*2^{24}$,$10*2^{24}$.Заметно удаление маловероятного потока в точках дороги и тротуара. Справа коэффициенты $1000 * 2^{24}$, $1000 * 2^{24}$.

(рис. 6)Точки с удовлетворяющие условию на величину максимального с.ч. для различных коэффициентов фильтрации. Слева 0.005%, справа 0,0001%.



(рис. 7) Изображение потока, вычисленного методом ViFlow. Слева изображен поток, без применения фильтра, справа после применения фильтра.



(рис.8) Пример вычисления потока на простейшем движении – параллельном переносе. Справа сверху пирамидальный Kanade-Lucas, высота 3,4 итерации метода Ньютона, слева сверху пирамидальный LK с фильтрацией, снизу ViFlow с применением фильтра.

По данным изображениям нельзя однозначно сказать, что какой то из методов явно лучше другого. Сравнение потоков на рис.5 и 7 показывает, что в обычной обстановке метод Канаде-Лукаса дает большую плотность потока чем ViFlow, при достаточно правдоподобном результате. В тоже время на рис.8 видно, что даже в простейшем случае метод Канаде-Лукаса может вычислять поток с большой неточностью, даже с использованием фильтрации, в то время поток вычисленный ViFlow в данном случае однороден и также имеет большую плотность. При этом функция уточнения для ViFlow выдает малые смешения для точек и ни на что не влияет.

Ниже представлено 2 таблицы, в которой показаны основные характеристики потоков, не зависящие от конкретных задач – плотность потока и скорость работы метода. В первой таблице приведены данные о потоке с двух последующих кадров видеоряда, во втором – с двух изображений, второе из которых получено параллельным переносом первого.

(табл.1)

	KLT, Высота 3, 4 итерации, к-ты фильтр. 10^3, 10^3, 10^3	KLT, Высота 3, 4 итерации, к-ты фильтр. 10^4, 10^4, 10^4	KLT, Высота 3, 4 итерации	KLT, Высота 2, 4 итерации	KLT, Высота 2, 4 итерации, к-ты фильтр. 10^4, 10^4, 10^4	ViFlow	ViFlow уточненный
Плотность, %	18,8	7,5	97	98,3	7,9	21,8	21,6
Время,clocks	2610	1187	11000	10280	1125	297	580

(табл.2)

	KLT, Высота 3, 4 итерации, к-ты фильтр. 10^3, 10^3, 10^3	KLT, Высота 3, 4 итерации, к-ты фильтр. 10^4, 10^4, 10^4	KLT, Высота 3, 4 итерации	KLT, Высота 2, 4 итерации	KLT, Высота 2, 4 итерации, к-ты фильтр. 10^4, 10^4, 10^4	ViFlow	ViFlow уточненный
Плотность, %	59,5	29,75	94,7	96,6	33,3	77	75
Время, clocks	7703	4234	11656	11297	4469	370	850

Направление дальнейшей работы

В данной работе предложен новый быстрый алгоритм поиска оптического потока с субпиксельной точностью, а также реализован уже существующий алгоритм Канаде-Лукаса и произведены их сравнения. Также при реализации получены некоторые методы, которые могут использоваться в дальнейшем (такие, как реализация создания пирамиды изображений).

При выполнении данной работы был получен относительно быстрый способ поиска потока для точек кадра(4 с. для изображения 640x480) который планируется применить в задаче восстановления 3D-сцены изображения(structure from motion) и задаче повышения разрешения последовательности изображений(superresolution)

В работе [4] рассмотрены некоторые алгоритмы поиска потока для задачи суперразрешения. В частности, лучшие результаты были достигнуты, используя метод Хорна-Шанка для поиска соответствий. Планируется сравнить качество полученных изображений и скорости работы в задаче распознавания номеров автомобилей, используя реализованный метод Канаде-Лукаса.

Локальные методы поиска потока более устойчивы к различным шумам и более точны при сопоставлении точек находящихся на границе объектов. Допускается, что применение иерархического КЛТ алгоритма для высоты пирамиды n=1,2 к небольшому участку изображения с номером автомобиля, даст лучшие результаты, чем методы, рассматривавшееся в работе [4].

В настоящее время проводятся эксперименты с методом, описанным в работе Бейкера и Канаде[5].

Суть алгоритма заключается в следующем:
1)Каждое из введенных изображений интерполируется методом бикубической интерполяции.
2)Для каждого изображения вычисляется оптический поток относительно N предыдущих и N последующих кадров.
3)Для каждого пикселя текущего изображения вычисляется среднее значение по соответствующим пикселям в N предыдущих и N последующих кадрах.

4)К полученному изображению применяется фильтр Винера.

В качестве метода поиска потока используется реализованный KLT алгоритм. Так как номер автомобиля имеет достаточно небольшой размер и четко выраженную границу, применение иерархического KLT метода должен давать относительно неплохие результаты. Еще одним достоинством при иерархического KLT метода является высокая плотность полученного потока, особенно в окрестности границ объектов.

Второй задачей в которой существует возможность применить полученные два метода, является задача восстановления трехмерной структуры сцены по изображениям этой сцены при разных углах и разных точек обзора.

Как правило, алгоритм восстановления сцены имеет следующие шаги
1)Выделение особых точек из изображений.
2)Сопоставление особенностей двух изображений.
3) Поиск решения и его уточнение
4) Вывод общего представления сцены
5)Вывод геометрических и других особенностей сцены.

Планируется применить один из полученных методов на первом и втором шагах для поиска соответствий между особенностями.
Предполагается, что используя один из полученных методов, может быть получен алгоритм восстановления сцены из водеопотока в реальном времени. Видео http://www.youtube.com/watch?v=9SUiC7ZR9PY&feature=related показывает довольно неплохой результат восстановления сцены, используя тот же пирамидальный метод. Канаде-Лукаса.

Список литературы

[1]. Форсайт Д., Понс Ж. Компьютерное зрение Современный подход//Издательский дом “Вильямс”

[2] Конушин А. Слежение за точечными особенностями сцены//Компьютерная графика и мультимедиа. Выпуск №1(5)/2003

[3]. Pyramidal Implementation of the Lucas-Kanade Feature Tracker

Description of the algorithm

[4] Attila Nagy, Zoltan Vamossy. Super-Resolution for Traditional and

Omnidirectional Image Sequences

[5] Simon Baker, Takeo Kanade. Super-resolution optical flow