Поиск оптимальных параметров алгоритма Lucas-Kanade

Ермолчев Алексей Юрьевич, «Научное предприятие «Цезис», лаборант
Свирин Илья Сергеевич, кандидат технических наук, ЗАО «Нордавинд»

Задача трекинга – определение местоположения движущегося объекта (нескольких объектов) во времени с помощью камеры, является одной из самых часто возникающих задач в системах компьютерного зрения. Системы трекинга обычно используют модель движения, которая описывает, как может изменяться изображение целевого объекта при всевозможных различных его движениях. При всей кажущейся сложности, на практике зачастую достаточно найти смещения двумерных проекций объектов в плоскости кадра. При необходимости проследить смещение объекта относительно его положения на предыдущем кадре, в первую очередь, поможет оптический поток (*optical flow*).

Описание алгоритма

Один из самых широко используемых дифференциальных методов оценки оптического потока является метод *Lucas-Kanade*, основанный на частных производных сигнала [1–2].

Приведем основное уравнение оптического потока:

$$\nabla I^T \cdot \vec{V} = -I_t$$

Уравнение содержит две неизвестных переменных и не может быть однозначно разрешено. Алгоритм *Lucas-Kanade* обходит неоднозначность за счет использования информации о соседних пикселях в каждой точке. Метод основан на предположении, что в локальной окрестности каждого пикселя *p* значение оптического потока одинаково; таким образом, можно записать основное уравнение оптического потока для всех пикселей окрестности и решить полученную систему уравнений методом наименьших квадратов. Алгоритм *Lucas-Kanade* менее чувствителен к шуму на изображениях, чем поточные методы, однако является сугубо локальным и не может определить направление движения пикселей внутри однородных областей.

Предположим, что смещение пикселей между двумя кадрами невелико. Рассмотрим пиксель *p*, тогда, по алгоритму *Lucas-Kanade*, оптический поток должен быть одинаков для всех пикселей, находящихся в окне с центром в *p*. А именно, вектор оптического потока (V_x, V_y) в точке *p* должен быть решением системы уравнений:

$$\begin{cases} I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2) \\ \dots \\ I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n) \end{cases}$$

где

$q_1, q_2, \dots, q_n$ — пиксели внутри окна,

$I_x(q_i), I_y(q_i), I_t(q_i)$ — частные производные изображения I по координатам *x*, *y* и времени *t*, вычисленные в точке q_i.

Это уравнение может быть записано в матричной форме: $Av = b$, где

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}, \qquad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \qquad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$$

Полученную переопределенную систему решаем с помощью метода наименьших квадратов. Таким образом, получается система уравнений 2×2:

$A^T Av = A^T b$, откуда $\mathrm{v} = (A^T A)^{-1} A^T b$,

где A^T — транспонированная матрица A. Получаем:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_x(q_i) I_y(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix}$$

Необходимым условием для работы с алгоритмом трекинга *Lucas-Kanade* является выбор параметров – порога фильтрации точек, количества уровней пирамид, размера окна поиска на каждом уровне пирамиды и критерия прекращения итерационного алгоритма поиска – при которых будет наблюдаться оптимальное соотношение ошибки вычисления и времени работы алгоритма.

В качестве алгоритма, используемого для проведения эксперимента — поиска оптимальных параметров — использовалась реализация алгоритма из библиотеки *opencv*. В данной библиотеке алгоритм *Lucas-Kanade* может быть запущен посредством вызова следующей *C++* функции:

```
void calcOpticalFlowPyrLK(
                InputArray prevImg,
                InputArray nextImg,
                InputArray prevPts,
                InputOutputArray nextPts,
                OutputArray status,
                OutputArray err,
                Size winSize=Size(21,21),
                int maxLevel=3,
                TermCriteria criteria=
                TermCriteria(
                        TermCriteria::COUNT+TermCriteria::EPS,
                        30, 0.01),
                int flags=0,
                double minEigThreshold=1e-4 )
```

Ниже приведено описание параметров, для которых в процессе проведения эксперимента предстоит выбрать оптимальные значения:

- *minEigThreshold*. Алгоритм находит минимальное собственное значение матрицы пространственных градиентов уравнения оптического потока, поделенное на количество пикселей в окне поиска, и если это значение меньше, чем *minEigThreshold*, то соответствующая точка отфильтровывается, и ее оптический поток не обрабатывается, что позволяет удалить «плохие» точки и получить повышение производительности.

- *maxLevel*. Если *maxLevel* установлен в 0, пирамиды не используется (один уровень), если установлен в 1, используется пирамида из двух уровней, и так далее.

- *winSize*. Размер окна поиска на каждом уровне пирамиды.

- *criteria*. Критерии прекращения итерационного алгоритма поиска (после указанного максимального числа итераций *criteria.maxCount*, или когда окно поиска станет меньше, чем *criteria.epsilon*).

Задача эксперимента — определить, при каких значениях входных параметров *minEigThreshold*, *maxLevel*, *winSize* и *criteria* данный алгоритм будет иметь минимальную ошибку вычисления и минимальное время обработки на заданном множестве возможных значений параметров. Так как требование ми-

нимальной ошибки и минимального времени обработки противоречивы, то необходимо оптимальное соотношение между этими величинами. Таким образом, рассматриваемая задача – является задачей многокритериальной оптимизации.

Сначала необходимо исказить изображения, чтобы создать выборку для исследования алгоритма трекинга. Алгоритм этих преобразований описан ниже:

1. Выбор 10 произвольных изображений, на каждом из которых случайным образом отмечена точка (её координаты записываются в файл).
2. Выбор одного изображения выборки.
3. К выбранному изображению последовательно применяются аффинные преобразования (масштабирование и параллельный перенос). В результате получается новое изображение. Новые координаты точки, отмеченной в п. 1 на исходном изображении, сохраняются в файл.
4. К полученному в предыдущем пункте изображению применяются аффинные преобразования. В результате получается новое изображение и новые координаты точки.
5. Процесс продолжается по аналогии еще 98 раз. Таким образом, получается 100 новых изображений (и отмеченных на них точек) для одного изображения исходной выборки
6. Выбор следующего изображения из исходной выборки и для него повторяются действия пунктов 3-5.

В итоге получилось 10 выборок по 100 изображений в каждой.

После применения аффинных преобразований вычисляются координаты точек: аналитически и с помощью алгоритм трекинга *Lucas-Kanade*. Для каждой пары точек ошибка вычислялась следующим образом: пусть $(x_1; y_1)$ — координаты точки, полученной с помощью аналитического расчёта, а $(x_2; y_2)$ – координаты точки, полученной с помощью алгоритма трекинга. Тогда ошибка d вычисляется как среднеквадратичное расстояние между аналитическим расчё-

том координат точки и расчётом координат этой же точки с помощью алгоритма *Lucas-Kanade*, по формуле

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

Ниже приведена общая последовательность действий по анализу значений параметров алгоритма трекинга:

1. Фиксируются параметры алгоритма трекинга.
2. Берется множество из 100 искаженных изображений, сгенерированных ранее и соответствующих одному из изображений исходной выборки, и для каждых двух последовательных изображений множества вычисляется ошибка и время работы алгоритма. В конце вычисляется средняя ошибка и среднее время работы алгоритма.
3. Шаг 2 повторяется для каждых 100 изображений, соответствующих изображениям исходной выборки. В итоге мы имеем 10 значений средней ошибки и 10 значений среднего времени работы алгоритма.
4. Усредняются средние значения, полученные на предыдущем шаге. Результаты заносятся в таблицу (см. Табл. 1).
5. Изменяется значение одного из параметров алгоритма (*minEigThreshold*, *maxLevel* или *winSize*), а затем повторяются шаги 2–5 до тех пор, пока все возможные сочетания параметров алгоритма (из заданных интервалов) не будут перебраны.

В данном эксперименте:

1. Параметр *minEigThreshold* изменялся от 0,1 до 0,001 с множителем 0,1.
2. Параметр *maxLevel* – от 1 до 4 с шагом 1.
3. Параметр *winSize* – от 3 до 59 с шагом 2 (брались только нечётные значения).

В табл. 1 приведена часть результатов, полученных в процессе анализа значений параметров алгоритма трекинга.

Таблица 1. Зависимость ошибки и времени работы алгоритма

трекинга от входных параметров

minEigThreshold	*maxLevel*	*winSize*	*average difference*	*average time*
0,1	1	3	157,4285000	0,0014998
0,1	1	5	141,2771300	0,0015417
0,1	1	7	65,5367240	0,0015830
0,1	1	9	31,4928630	0,0015702
0,1	1	11	24,5364290	0,0016121
0,1	1	13	21,0288450	0,0016513
0,1	1	15	26,4307920	0,0016966
0,1	1	17	11,8516820	0,0017262
0,1	1	19	12,0752020	0,0017485

Из данной таблицы необходимо было выбрать оптимальные параметры. Для этого был использован *критерий Парето (*Предположим, что рассматриваются два набора параметров, *M* и *N*. Говорят, что *M* доминирует *N* по Парето, если *M* не хуже *N* по всем критериям и хотя бы по одному критерию превосходит *N*. Если так оно и есть на самом деле, то, действительно, в выборе *N* нет никакого смысла. Ведь *M* по всем параметрам не уступает, а по каким-то и выигрывает *N* [3]).

В соответствии с данными критерием были выбраны следующие параметры:

minEigThreshold=0,001,

maxLevel=3,

winSize=(21; 21).

Для данных параметров с целью определить влияние параметра *criteria* на ошибку и среднее время работы алгоритма был проведён еще один эксперимент (см. Рис. 1-4).

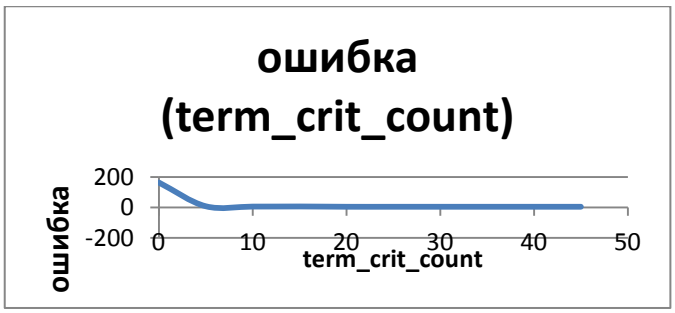

Рис. 1. Зависимость ошибки алгоритма от параметра *term_crit_count*

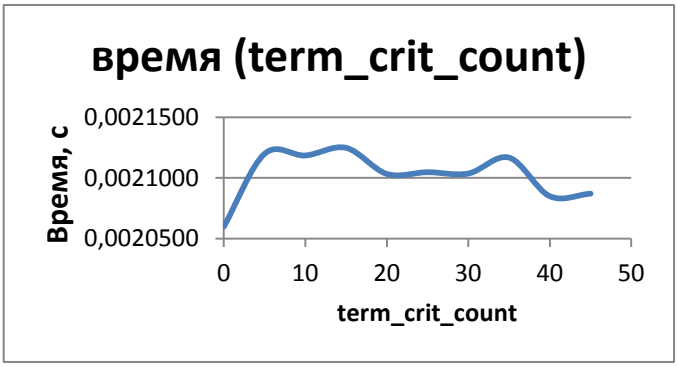

Рис. 2. Зависимость времени работы алгоритма от параметра *term_crit_count*

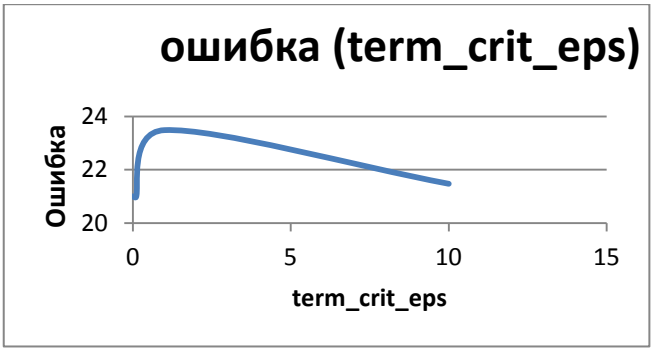

Рис. 3. Зависимость ошибки алгоритма от параметра *term_crit_eps*

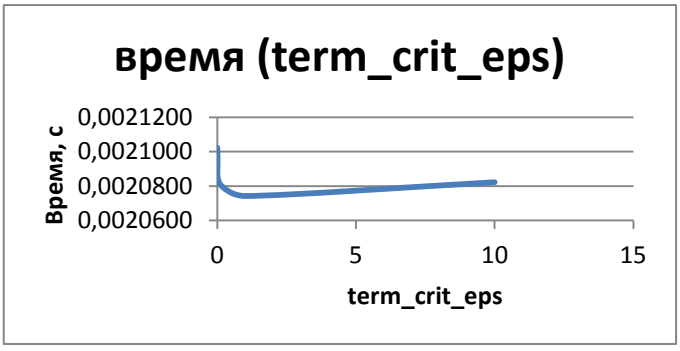

Рис. 4. Зависимость времени работы алгоритма от параметра *term_crit_eps*

На рис.1 видно, что при значении параметра *term_crit_count* больше 5 величина ошибки стремится к нулю, тогда как закономерность изменения времени работы алгоритма от этого параметра (рис. 2) определить трудно.

На рис. 3 показано, как изменяется ошибка в зависимости от изменения параметра *term_crit_eps* и видно, что при значении параметра больше 2 значение ошибки убывает, тогда как на рис. 4, где отображена зависимость времени работы алгоритма от этого же параметра, после значения 2 время, наоборот, возрастает.

Таким образом, для решения задачи, описанной в данной работе, был проведён эксперимент, в ходе которого и были выявлены необходимые значения параметров алгоритма трекинга *Lucas-Kanade*, а для параметра *criteria* были построены графики, на которых можно увидеть зависимости ошибки и времени работы алгоритма от этого параметра.

Список литературы

1. Лукьяница А.А. Цифровая обработка видеоизображений / А.А. Лукьяница, А.Г. Шишкин. – М.: Ай-Эс-Эс Пресс, 2009. – 518 с.

2. Lucas B.D. An Iterative Image Registration Technique with an Application to Stereo Vision / B.D. Lucas, T. Kanade // Proceedings of the 7th international joint conference on Artificial intelligence. – 1981. – Vol. 2. – P. 674–679.

3. Перевод раздела из книги Luke S. Essentials of Metaheuristics. A Set of Undergraduate Lecture Notes. Zeroth Edition. Online Version 0.5. October, 2009 (http://cs.gmu.edu/~sean/book/metaheuristics/).

4. Солодова Е.Н., Епишина Е.В., Набильская Н.В. Анализ рынка облачного сервиса интеллектуальной обработки видеопотоков высокой доступности, интегрированного с корпоративными и ведомственными видеоподсистемами, основанных на использовании открытых

протоколов.// Проблемы экономики и менеджмента, №2, 2015, с.-132-135.

5. Солодова Е.Н., Епишина Е.В. Влияние регионального университета на малое инвестиционное предприятие (на примере Международного университета природы, общества и человека «Дубна» и ООО «Научное предприятие «Цезис») // Сборник научных работ II международной научной конференции Евразийского Научного Объединения (г. Москва, февраль 2015). — Москва: ЕНО, 2015. — 350 с.