УДК 535.31

АЛГОРИТМЫ ПОЗИЦИОНИРОВАНИЯ МОБИЛЬНОГО УСТРОЙСТВА НА ОСНОВЕ ДАННЫХ ОТ ВСТРОЕННОЙ ФОТОКАМЕРЫ

А.В. Козырева

Институт систем информатики им. А.П. Ершова СО РАН, г. Новосибирск
E-mail: alina.kozyreva@gmail.com

Изучена возможность создания программного продукта для позиционирования мобильного устройства в пространстве относительно окружающих объектов. Рассмотрены алгоритмы калибровки камеры, а также алгоритмы нахождения особенностей изображения и их дальнейшего отслеживания. Определение того, изменилось ли положение в пространстве или нет, производится путем поиска неизвестных коэффициентов в матрице вращения и векторе сдвига на основе полученных калибровочных данных и найденных особенностей. На основе результатов работы разных алгоритмов создана упрощенная техника для решения конкретных задач. Результаты ее работы показали неплохие результаты. Данные сочетания техник могут быть использованы для управления мобильными устройствами или в играх.

Ключевые слова:
Калибровка камеры, особенности изображения, угловые детекторы, алгоритмы слежения за особенностями.

Key words:
Camera calibration, feature detection, edge detection, feature point tracking.

Введение

В работе идет речь о возможности создания программного продукта для позиционирования мобильного устройства (смартфон или КПК) в пространстве относительно окружающих объектов на основе данных, полученных от установленной на нем фотокамеры. На данный момент аналогичные приложения существуют. Но они или разработаны для более мощных устройств, нежели мобильный телефон или КПК, или устройство обладает датчиками, способными решать данную проблему на основе других физических принципов. Данная работа рассматривает алгоритмы, позволяющие создать приложения, которые могли бы работать на большинстве мобильных устройств вне зависимости от их производителя. Хотя можно отметить, что цели такой разработки не совсем ясные, и это, в значительной мере, является экспериментом.

Поставленную задачу предлагается решать следующим образом. Найдя особые точки на первом изображении, необходимо отыскать их на втором. Здесь потребуются алгоритмы слежения за особенностями (опорными точками). Для решения задачи используем минимум четыре точки на одном изображении и соответствующие четыре точки на втором. Так как опорные точки на получаемых изображениях имеют две координаты, а соответствующая точка пространства – три координаты, то необходимо знать правила, по которым происходит данное преобразование. Эти правила зависят от внутренних параметров фотокамеры. Их определение осуществляется посредством калибровки камеры. Определение того, изменилось ли положение в пространстве или нет, производится путем поиска неизвестных коэффициентов в матрице вращения и векторе сдвига.

1. Процесс калибровки камеры

Рассмотрим рис. 1. Точка $m=(x,y)$ в плоскости камеры и точка $M=(X, Y, Z)$ в пространстве связаны следующим равенством:

$$Z\mathbf{v}=A\mathbf{M}, \qquad (1)$$

где $\mathbf{M}=(X, Y, Z)^T$ – трехмерный вектор, соответствующий точке M, $\mathbf{m}=(x,y)^T$ – двумерный вектор, соответствующий точке m, $\mathbf{v}=(u\ v\ 1)^T$ – вектор однородных [1] внутренних координат камеры, а матрица $A=\begin{bmatrix} f/w & \gamma & u_0 \\ 0 & f/h & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ – матрица, известная под названием матрицы внутренних параметров камеры, поскольку она содержит только параметры оптической системы и фотоприемника камеры.

Пусть задана точка $m=(x,y)$ на плоскости, которая является проекцией некоторой точки $M=(X,Y,Z)$ в пространстве. Соотношение между этими двумя точками можно представить в следующем виде:

$$sm=A[R\ t]M, \qquad (2)$$

где s – некоторый скаляр, пара $(R\ t)$ – так называемые внешние параметры камеры, A – матрица, представляющая собой внутренние параметры камеры. Получим новое соотношение из (2), использовав матрицу гомографии $H=A[r_1\ r_2\ t]$: $sm=HM$. Матрицу гомографии можно вычислить различными способами [2], будем считать, что она уже есть. Положим, что $H=[h_1\ h_2\ h_3]$, тогда из (2) получим:

$$[h_1 \quad h_2 \quad h_3] = \lambda A[r_1 \quad r_2 \quad t],$$

где λ – скаляр. Исходя из того, что r_1 и r_2 ортогональны, получаем

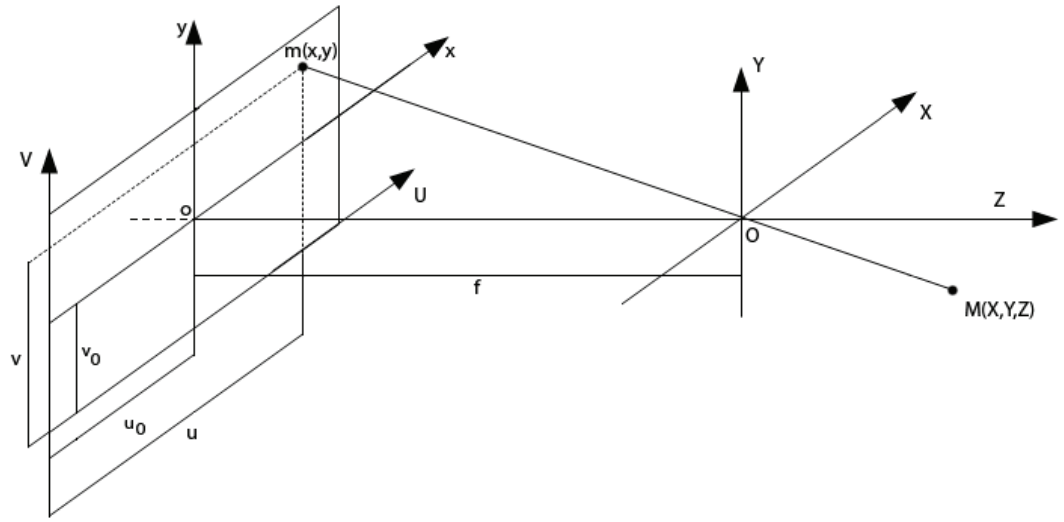

Рис. 1. *Система координат проективной камеры*

$$h_1^T A^{-T} A^{-1} h_2 = 0; \tag{3}$$

$$h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2. \tag{4}$$

Положим

$$v_{ij} = \begin{bmatrix} h_{i1}h_{j1},\ h_{i1}h_{j2} + h_{i2}h_{j1},\ h_{i2}h_{j2}, \\ h_{i3}h_{j1} + h_{i1}h_{j3},\ h_{i3}h_{j2} + h_{i2}h_{j3},\ h_{i3}h_{j3} \end{bmatrix}^T,$$

тогда (3) и (4) можно переписать в следующем виде:

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} \mathbf{b} = 0. \tag{5}$$

В (5) $\mathbf{b}$ – это 6D вектор, равный $[B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]$, где B_{ij} – элементы матрицы $B = A^{-T}A^{-1}$.

Тогда, используя n изображений шаблона, путем несложных преобразований [3] элементы матрицы внутренних параметров можно записать следующим образом:

$$v_0 = (B_{12}B_{13} - B_{11}B_{23}) / (B_{11}B_{22} - B_{12}^2),$$

$$\lambda = B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})] / B_{11},$$

$$\alpha = \sqrt{\lambda / B_{11}},$$

$$\beta = \sqrt{\lambda B_{11} / (B_{11}B_{22} - B_{12}^2)},$$

$$\gamma = -B_{12}\alpha^2 \beta / \lambda,$$

$$u_0 = cv_0 / \alpha - B_{13}\alpha^2 / \lambda,$$

где B_{ij} – элементы матрицы $B = A^{-T}A^{-1}$.

Калибровку стационарной камеры можно провести также следующим образом [8]. Пусть даны перекрывающие друг друга изображения $J_0 \ldots J_N$, где $N \geq 2$. Изображения получены при помощи одной камеры и регистрируют одну и ту же сцену. Тогда алгоритм самокалибровки следующий:

- установить соответствие между точками изображений;
- для каждого $j = 1 \ldots N$ вычислить преобразование P_j между J_0 и J_i;
- найти верхнетреугольную матрицу K, такую что $K^{-1}P_jK = R_j$, где R_j – это матрица вращения для всех $j > 0$ (K – калибровочная матрица, R_j описывает ориентацию камеры в момент j по отношению к «нулевому» моменту);
- итеративно улучшить полученную оценочную калибровочную матрицу.

2. Вращение и сдвиг

Рассмотрим общий случай, когда оптические оси камер не параллельны, а направление смещения оптического центра одной камеры относительно оптического центра другой произвольно (рис. 2). Введем для каждой камеры свою стандартную систему координат. Пусть первой камере соответствует система координат $O'X'Y'Z'$, а второй – $O''X''Y''Z''$ (рис. 2). Пусть вектор $\mathbf{M}' = (O'X'Y'Z')^T$ характеризует координаты некоторой точки M трехмерного пространства в системе первой камеры, а вектор $\mathbf{M}'' = (O''X''Y''Z'')^T$ – в системе второй. Легко показать, что связь между векторами $\mathbf{M}'$ и $\mathbf{M}''$ задается соотношением

$$\mathbf{M}'' = R\,\mathbf{M}' + t, \tag{6}$$

где $R = R''R'^T$ – ортогональная матрица, описывающая ориентацию системы координат второй камеры относительно первой, а $t = -R''R'^T t' + t''$ – вектор трансляции, определяющий положение оптического центра второй камеры в системе координат первой.

Рассмотрим пару камер, внутренние параметры которых известны, но неизвестны внешние параметры (матрица R и вектор t). Умножив обе части выражения (6) слева векторно на t, а затем скалярно – на $\mathbf{M}''$, получим $M''^T(t \times RM') = 0$. Это соотношение формально выражает тот факт, что векторы $\mathbf{M}'$, $\mathbf{M}''$ и t лежат в одной плоскости, проходящей через три точки: оптические центры камер O' и O'' и точку наблюдения M. Выражая M через v и учитывая свойства смешанного произведения векторов, получим:

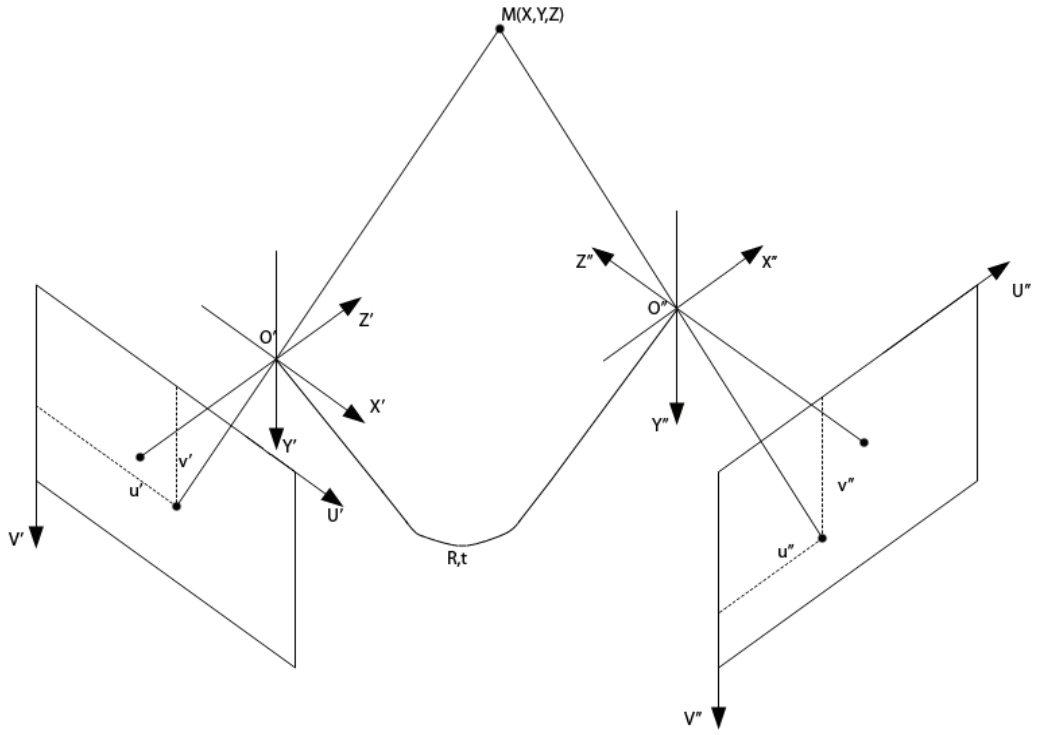

Рис. 2. *Система двух произвольно ориентированных камер*

$$(v''^{T} A_2^{-T} \times R A_1^{-1} v')^{T} t = 0. \qquad (7)$$

Соотношение (7) является основой для оценки матрицы R и вектора t. Предположим, что известны координаты n пар сопряженных точек и, соответственно, n пар векторов v_i' и v_i'', i=1, 2... n.

Рассмотрим метод оценивания R и t, использующий (7). Так как это соотношение справедливо для любой пары сопряженных точек, имеем систему из n уравнений относительно неизвестных R и t, которую можно представить в виде:

$$B_n t = 0, \text{ где } B_n = \begin{bmatrix} (v_1''^{T} A_2^{-T} \times R A_1^{-1} v_1')^{T} \\ \dots \\ (v_n''^{T} A_2^{-T} \times R A_1^{-1} v_n')^{T} \end{bmatrix}. \qquad (8)$$

Система (8) является однородной линейной по t. Это означает, что вектор трасляции можно оценить только с точностью до постоянного множителя. Вводя условие нормировки $\|t\|^2$=1, количество возможных решений можно ограничить двумя, отличающимися знаком. Система (8) содержит пять неизвестных, следовательно, и число пар известных сопряженных точек n должно быть не менее пяти.

Поскольку на практике в матрицу B_n входят не точные значения координат сопряженных точек, а результаты их измерений, которые могут содержать ошибки, то реально система (8) имеет ненулевую правую часть, т. е. $B_n t$=**e**, где **e** – вектор невязки, обусловленный наличием ошибок измерений.

Согласно МНК в качестве оценок матрицы вращения и вектора трансляции следует выбрать такие R и t, которые минимизируют значения функционала $J_2(R,t)=e^{T}e=t^{T}B_n^{T}B_n t$. Как упоминалось ранее, при условии $\|t\|^2$=1 квадратичная форма $t^{T}B_n^{T}B_n t$ достигает минимума $J_2=\lambda_{\min}$ по t ($\lambda_{\min}$ – минимальное собственное число матрицы $B_n^{T}B_n$), если t – собственный вектор матрицы, соответствующий $\lambda_{\min}$. Поэтому процедуру оценивания R и t можно разбить на два этапа. На первом – находится матрица R, минимизирующая $\lambda_{\min}$. На втором – оценивается собственный вектор матрицы $B_n^{T}B_n$, соответствующий $\lambda_{\min}$. Существует множество алгоритмов и их программных реализаций для вычисления собственных векторов, поэтому второй этап не вызывает трудностей.

Значительно более сложной задачей является задача оценивания матрицы R. Один из возможных алгоритмов состоит в следующем. Известно, что матрица R может быть представлена в виде $R=R_x(\omega_x)R_y(\omega_y)R_z(\omega_z)$, где

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega_x & -\sin\omega_x \\ 0 & \sin\omega_x & \cos\omega_x \end{bmatrix},$$

$$R_y = \begin{bmatrix} \cos\omega_y & 0 & \sin\omega_y \\ 0 & 1 & 0 \\ -\sin\omega_y & 0 & \cos\omega_y \end{bmatrix},$$

$$R_z = \begin{bmatrix} \cos\omega_z & -\sin\omega_z & 0 \\ \sin\omega_z & \cos\omega_z & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Углы ω_x, ω_y и ω_z – три неизвестных параметра, от которых зависит матрица R. На практике всегда известен диапазон, в котором они могут лежать. Выполняя в этом диапазоне полный перебор по всем углам с достаточно грубым шагом (например, 1°), можно приблизиться к значениям, удовлетворяющим требованиям минимизации функционала J_2 по R. Затем в окрестности этих значений для уточнения положения минимума можно воспользоваться одним из известных методов минимизации (например, наискорейшего спуска, Ньютона, Маркуардта).

3. Опорные точки изображения

Введем понятие точечной особенности x' изображения, как точки, чья окрестность отличается от окрестностей близлежащих точек по выбранной мере, т. е. $\{\forall x: |x'-x|<r \rightarrow \rho(\Omega_x, \Omega_{x'})>\varepsilon\}$, где Ω_x – окрестность точки x, называемая окном поиска, а $\rho(\Omega_x, \Omega_{x'})$ – функция близости окрестностей по некоторой мере.

Большинство детекторов точечных особенностей работают сходным образом: для каждой точки изображения вычисляется некоторая функция от ее окрестности. Точки, в которых эта функция достигает локального максимума, очевидно можно отличить от всех точек из некоторой ее окрестности.

Существует целый набор функций, которые можно использовать для обнаружения точечных особенностей. Чаще всего для задач отслеживания точек сцены применяются функции, находящие в изображении структуры, похожие на угол – уголки (*corners*). Детекторы, использующие такие функции, называются детекторами углов. Именно они чаще всего применяются для решения задач отслеживания точечных особенностей сцены.

3.1. Алгоритм Харриса и Стефенса/Плесси

Угловой детектор Харриса основывается на местной функции сигнала. Пусть у нас есть точка (x,y) и некоторое смещение $(\Delta x,\Delta y)$, тогда функцию сигнала можно записать следующим образом:

$$c(x, y) = \sum_W [I(x_i, y_i) - I(x_i + \Delta x, y_i + \Delta y)]^2, \qquad (9)$$

где I является функцией изображения, а (x_i,y_i) – точка в окне W с центром в точке (x,y). Разложив функцию I по Тейлору и подставив в (9), получим: $c(x, y) = [\Delta x\ \Delta y] C(x, y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$, где $C(x,y)$ фиксирует яркость в локальной окрестности.

Пусть λ_1, λ_2 являются собственными значениями матрицы $C(x,y)$. Возможны три варианта.

1. Если оба значения λ_1 и λ_2 близки к 0, то это означает, что функция яркости в окрестности данной точки меняется незначительно, а значит, она не может быть точкой особенности.
2. Если одно собственное значение матрицы большое, а другое близко к нулю, то локальная функция яркости меняется в одном направлении; это «ребро».
3. Если оба значения высоки, то даже небольшое смещение точки (x, y) в сторону вызывает значительные изменения в яркости, что может быть особенностью изображения; это «угол».

Точки изображения, соответствующие локальным максимумам функции $R=\det C - k(\mathrm{trace} C)^2$, и признаются особенностями. Параметр k обычно полагается равным 0,04...0,15 (предложено Харрисом).

Для снижения влияния шумов на найденные особенности используется сглаживание по Гауссу, но не в самом изображении, а в картах частных производных.

Во многих случаях находится чересчур большое количество углов, из-за чего в дальнейшем их будет сложно отслеживать. Поэтому вводится ограничение на минимальное расстояние между найденными особенностями, и все лишние отбрасываются.

3.2. Угловой детектор SUSAN

Название SUSAN происходит от сокращения полного английского названия алгоритма: Smallest Univalue Segment Assimilating Nucleus.

Рассмотрим рис. 3 с изображенным темным прямоугольником на белом фоне. Круговая маска (с центром в точке, которую будем называть ядром) показана на рисунке в пяти возможных позициях. Если яркость каждого пикселя внутри маски сопоставима с яркостью ядра данной маски, то тогда область маски может быть определена как имеющая одинаковую (или очень близкую) яркость с ядром. Эта часть маски называется USAN, сокращенно от Univalue Segment Assimilating Nucleus, и закрашена на рисунке белым цветом.

Для обнаружения особенностей SUSAN использует круглую маску с центром в «подозрительной» точке, называемой ядром. Радиус маски обычно равен 3,4 пикселя, что дает маску из 37 точек. Пусть M – некоторая область маски, $\bar{m}$ – точка, принадлежащая данному региону, а $\bar{m}_0$ – ядро маски. Будем сравнивать яркость каждой точки маски с ядром, используя следующую функцию:

$$c(\bar{m}, \bar{m}_0) = \begin{cases} 1 & if\ |I(\bar{m}) - I(\bar{m}_0)| \le t \\ 0 & if\ |I(\bar{m}) - I(\bar{m}_0)| > t \end{cases}. \qquad (10)$$

На основе данной функции, пройдя по всей маске, получим суммирующую функцию:

$$n(\bar{m}) = \sum_{\bar{m}} c(\bar{m}, \bar{m}_0). \qquad (11)$$

Эта функция обозначает число точек в области USAN. Параметр t в уравнении (10) показывает минимальную границу контраста для выделения особенности.

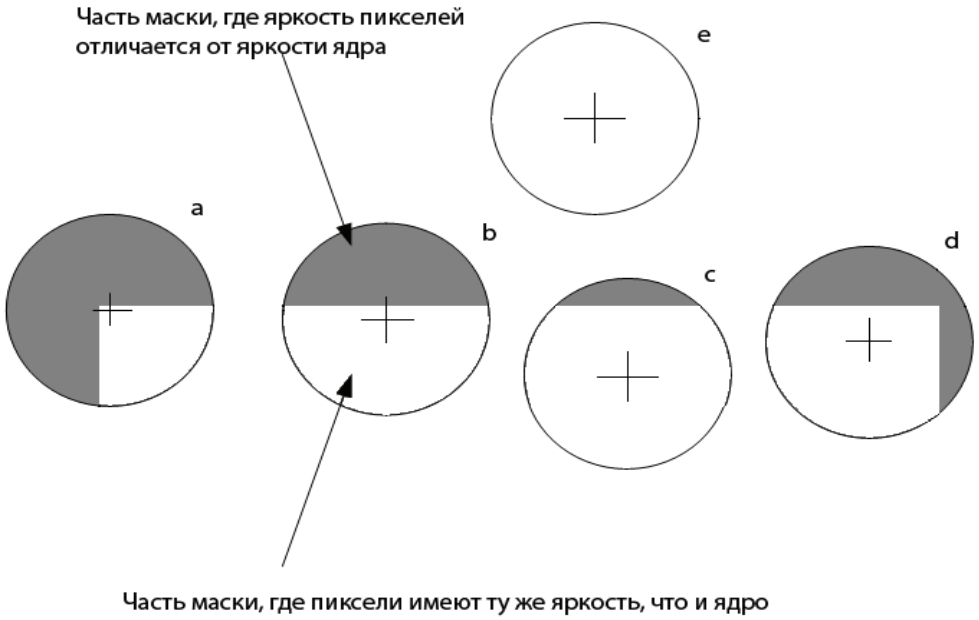

Рис. 3. *Круговые маски*

Далее, значение функции (11) в каждой точке (ядра) сравнивается с фиксированным порогом g (геометрическим порогом). Значение этого порога рекомендуется устанавливать равным $3n_{\max}/4$, где $n_{\max}$ – максимальное значение функции $n(\bar{m})$ на изображении.

Тогда SUSAN оператор вычисляется по следующей формуле:

$$R(\bar{m}_0) = \begin{cases} g - n(\bar{m}_0) & if\ n(\bar{m}_0) < g \\ 0 & if\ n(\bar{m}_0) \geq g \end{cases}. \qquad (12)$$

Значение оператора всегда положительное. Чем меньше USAN область, тем больше отклик угла.

Для более стабильного результата есть смысл заменить формулу (11) на следующую:

$$c(\bar{m}, \bar{m}_0) = e^{-\left(\frac{I(\bar{m}) - I(\bar{m}_0)}{t}\right)^6}. \qquad (13)$$

Те точки, в которых оператор SUSAN принимает не нулевое значение, могут быть точками особенности.

3.3. Угловой детектор Хэдли и Трайкович

Этот детектор особенностей состоит из точек в так называемом круге Брезенхама [10] радиуса r с центром в предполагаемой точке особенности. Если непрерывная последовательность из n точек ярче, чем некое ядро, на некоторое число t или наоборот, темнее, то тогда точка ядра будет особенностью изображения. Радиус r может быть любым, однако лучше использовать значение, равное 3 (соответствует кругу из 16 точек окружности). Кроме того, экспериментально вычислено, что наилучшее значение n – это 9. При n меньше 9 – углы не обнаруживаются на изображении. Детектор прямо проверяет точку с окружающим ее пространством. Если c – рассматриваемая точка, $p \in P$ – точка в круге P с центром вокруг c, а точка p' противоположна p по линии диаметра, то функция отклика выглядит следующим образом:

$$r(c) = \min_{p \in P}[(I(p) - I(c))^2 + (I(p') - I(c))^2].$$

Точки, в которых значение функции $r(c)$ будет больше заданного порога, определяются как потенциальные углы. Вычислить подлинные углы можно, использовав подавление в точках отсутствия максимума (non-maximum Suppression).

3.4. Развитие алгоритмов слежения

Все современные алгоритмы слежения за особенностями опираются на работу 1981 г. Лукаса и Канаде. В 1991 г. математическая формулировка этого алгоритма была изменена и стала основой для всех последующих обобщений с учетом аффинных искажений окрестности и освещенности.

- Lucas-Kanade – особенность считается только смещающейся, без искажений [5];
- Tomasi-Kanade – переформулирование Lucas-Kanade. Движение считается смещением, и рассчитывается путем итеративного решения построенной системы линейных уравнений [6];
- Shi-Tomasi-Kanade – учитывает аффинные искажения особенности [4].

3.4.1. Алгоритм Lucas-Kanade

Этот алгоритм в принципе применим для функций любой размерности n. Пусть x – особенность первой функции F. Необходимо найти такую точку $x+h$ функции G, что разность окрестностей этих точек по мере минимальна.

Расстояние между окрестностями записывается в виде $E = \sum_{x \in R} [F(x+h) - G(x)]^2$, где $F(x)$, $G(x)$ – две функции, которые можно разложить в ряд Тейлора. Используя это приближение, ищем минимум E путем дифференцирования и приравнивания производной к нулю. Смещение h можно получить как

$$h = \left[\sum_x \left(\frac{\partial F}{\partial x} \right)^T [G(x) - F(x)] \right] \left[\sum_x \left(\frac{\partial F}{\partial x} \right)^T \left(\frac{\partial F}{\partial x} \right) \right]^{-1}.$$

Как было указано ранее, задача слежения за особенностями без учета аффинных искажений является поиском величины оптического потока в наборе точек. Поэтому метод Lucas-Kanade часто применяется для поиска оптического потока во всем изображении.

3.4.2. Алгоритм Tomasi-Kanade

В этом алгоритме движение особых точек также описывается смещением вида: delta(x)=x+d. Как и в предыдущем алгоритме, задача заключается в поиске такого d, при котором минимизируется разность окон особенностей: $\varsigma = \sum_W [I(x+d, t+\tau) - I(x,t)]^2$. Разложив функцию изображения в ряд Тейлора, разницу между окнами по мере можно переписать в виде $\varsigma \approx \sum_W [\nabla I(x,t)^T d + I_t(x,t)\tau]^2$. Продифференцировав это выражение по d и приравняв производную к нулю, получаем линейную систему относительно d: $Cd=g$, где

$$C = \sum_W \begin{bmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{bmatrix} \quad u \quad g = -\tau \sum_W I_t [I_u I_v]^T.$$

Из этой системы получаем d как $d_k = C^{-1} g$.

С учетом приближения функции изображения с помощью ряда Тейлора, решение получается неточным. Для его уточнения удобно применить итеративную процедуру Ньютона–Рафсона.

3.4.3. Алгоритм Shi-Tomasi-Kanade

В этом алгоритме учитываются аффинные искажения изображения окрестности особых точек, поэтому движение пикселей окна особенности описывается в виде $Ax+d$, где A – матрица 2×2, а d – смещение.

Задача слежения за особенностью сводится к проблеме определения параметров движения и искажения окна особенности, при которой минимизируется разность $\varsigma = \iint_W [J(\Delta x + d) - I(x)]^2 w(x) dx$, где W – окно особенности, а w – весовая функция в окне, $J(x)$ и $I(x)$ – два изображения.

Выражение дифференцируется относительно параметров движения, и производная приравнивается к 0. Затем система линеаризуется с помощью разложения функции изображения в ряд Тейлора. Это дает нам линейную 6×6 систему, которая решается итеративно по методу Ньютона–Рафсона.

4. Решение задачи и результаты

Предположим, что имеются два снимка какой-то сцены. Необходимо узнать, как изменился второй снимок относительно первого. Были опробованы все описанные выше методы. Приемлемое по качеству решение было получено на следующем пути.

Сначала будем искать опорные точки, т. е. такие точки изображения, которые можно отличить от всех соседних с ней точек изображения. Поделим изображение на 9 одинаковых частей, и в каждой части будем искать опорную точку. Таким образом найдем наиболее удаленные друг от друга точки. Для поиска будем использовать алгоритм Трайковича. Для второго изображения воспользуемся алгоритмом Shi-Tomasi-Kanade по причине возможного аффинного преобразования. Таким образом, получили в лучшем случае два набора по девять точек. Это избыточно, но дает шанс для более точных дальнейших вычислений.

Есть уравнения вида:

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = R \begin{pmatrix} U_i \\ V_i \\ W_i \end{pmatrix} + t, \quad \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = K^{-1} \begin{pmatrix} x_i \\ y_i \end{pmatrix},$$

$$\begin{pmatrix} U_i \\ V_i \\ W_i \end{pmatrix} = K^{-1} \begin{pmatrix} u_i \\ v_i \end{pmatrix}, \quad i = 1...4,$$

где $(x_i,y_i)^T$ – точка в плоскости первого изображения, являющаяся проекцией точки в пространстве $(X_i,Y_i,Z_i)^T$, $(u_i,v_i)^T$ – это точка в плоскости второго изображения, являющаяся проекцией точки $(U_i,V_i,W_i)^T$ в пространстве, K^{-1} – матрица внутренних параметров камеры, R и t – искомые матрица вращения и вектор сдвига соответственно. Каждое уравнение указано в «своих» координатах: первое – в глобальной системе, второе – в системе камеры в первом положении, третье в системе камеры, находящейся во втором положении.

С учетом того, что известны двумерные координаты точки на каждом изображении и матрица внутренних параметров, можем считать, что и трехмерные координаты также известны. Таким образом, получаем четыре уравнения вида

$$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = R \begin{pmatrix} U_i \\ V_i \\ W_i \end{pmatrix} + t.$$

Распишем их более подробно:

$$\begin{cases} X_1 = r_{11}U_1 + r_{12}V_1 + r_{13}W_1 + t_1 \\ X_2 = r_{11}U_2 + r_{12}V_2 + r_{13}W_2 + t_1 \\ X_3 = r_{11}U_3 + r_{12}V_3 + r_{13}W_3 + t_1 \\ X_4 = r_{11}U_4 + r_{12}V_4 + r_{13}W_4 + t_1 \end{cases},$$

$$\begin{cases} Y_1 = r_{21}U_1 + r_{22}V_1 + r_{23}W_1 + t_2 \\ Y_2 = r_{21}U_2 + r_{22}V_2 + r_{23}W_2 + t_2 \\ Y_3 = r_{21}U_3 + r_{22}V_3 + r_{23}W_3 + t_2 \\ Y_4 = r_{21}U_4 + r_{22}V_4 + r_{23}W_4 + t_2 \end{cases}$$

$$\text{и} \begin{cases} Z_1 = r_{31}U_1 + r_{32}V_1 + r_{33}W_1 + t_3 \\ Z_2 = r_{31}U_2 + r_{32}V_2 + r_{33}W_2 + t_3 \\ Z_3 = r_{31}U_3 + r_{32}V_3 + r_{33}W_3 + t_3 \\ Z_4 = r_{31}U_4 + r_{32}V_4 + r_{33}W_4 + t_3 \end{cases}.$$

Видим, что неизвестные каждой системы не зависят от неизвестных других систем. В каждой системе четыре неизвестных: три из матрицы вращения и одна неизвестная из вектора сдвига. Поэтому и требовалось лишь четыре особых точки. Каждая система представляет линейные уравнения. Распишем более подробно первую систему:

$$\begin{cases} X_1 = r_{11}U_1 + r_{12}V_1 + r_{13}W_1 + t_1 \\ X_2 = r_{11}U_2 + r_{12}V_2 + r_{13}W_2 + t_1 \\ X_3 = r_{11}U_3 + r_{12}V_3 + r_{13}W_3 + t_1 \\ X_4 = r_{11}U_4 + r_{12}V_4 + r_{13}W_4 + t_1 \end{cases} \Rightarrow A \begin{pmatrix} r_{11} \\ r_{12} \\ r_{13} \\ t_1 \\ -1 \end{pmatrix} = 0,$$

где $A = \begin{pmatrix} U_1 & V_1 & W_1 & 1 & X_1 \\ U_2 & V_2 & W_2 & 1 & X_2 \\ U_3 & V_3 & W_3 & 1 & X_3 \\ U_4 & V_4 & W_4 & 1 & X_4 \end{pmatrix}$.

Приведя A к верхне-треугольному виду, получим решение в общем виде:

$$t_i = -A_{45}$$
$$r_{i3} = -A_{35} - t_i$$
$$r_{i2} = -A_{23}r_{i3} - t_i - A_{25}$$
$$r_{i1} = A_{12}r_{i2} - A_{13}r_{i3} - t_i - A_{15},\ i = 1...3.$$

Время работа алгоритмов проверялось на трех изображениях, заданных заранее. Изображения комбинировались различными способами, и замерялось время работы приложения. Среднее время работы оказалось равным 515,5 мс. Заметим, что аналогичная задача в литературе не рассматривалась, поэтому непосредственно сравнить эффективность нашего решения с подобной задачей невозможно. Однако, можно попытаться позиционировать нашу задачу среди аналогичных. Близкие, но более сложные задачи возникают при обработке космических снимков и в других областях. Часто они имеют неприемлемо большие времена решения, десятки минут. Для того, чтобы их решить за приемлемое время, применяют параллельные вычислительные системы. С другой стороны, аналогичные задачи встречаются в робототехнике. Требования на время реакции таких систем – десятки миллисекунд. В реальных системах эти параметры достигаются, но при этом исключаются аффинные преобразования, используют только параллельный перенос, поворот и изменение масштаба.

5. Заключение

В значительной мере данная разработка является экспериментальной. Тестирование алгоритмов проводилось в условиях близких у идеальным. Не были учтены некоторые условия. Полученные с камеры снимки могут отличаться резкостью, это может происходить по разным причинам: не успела произойти фокусировка, или фокус сместился относительно предыдущего объекта фокусировки. В связи с этим, возникает проблема определения соответствия точечных особенностей между кадрами со значительно отличающейся резкостью изображения. Традиционные методы, как сопоставления особенностей между кадрами, так и отслеживания перемещения особенностей, не справляются с подобной ситуацией.

Может меняться и степень освещенности изображения. В этом случае также возможно использовать метод «усреднения» изображения, т. е. приводить все полученные с камеры изображения к заранее определенному образцу. Один из таких методов описан в [5] и [7]. Как и в любой работе с изображениями, мы не застрахованы от наличия шумов на них, избавиться от которых можно при помощи специальных фильтров. Множество таких фильтров описаны в [9]. Но любое дополнительное действие над изображениями будет замедлять общую работу.

Разработанные алгоритмы можно использовать при разработке систем управления мобильным устройством, а также при создании разнообразных игр для мобильных устройств.

Рис. 4. *Изображения одной и той же сцены с трех разных ракурсов*

СПИСОК ЛИТЕРАТУРЫ

1. Грузман И.С., Киричук В.С., Косых В.П., Перетягин Г.И., Спектор А.А. Цифровая обработка изображений в информационных системах. – Новосибирск: Изд-во НГТУ, 2000. – 168 с.
2. Harker M., O'Leary P. Computation of Homographies // Austria, Leoben: Tech. Rep., Dep. of Product Engineering, Univ. of Leoben, 2005. – 10 p.
3. Zhengyou Z. A Flexible New Technique for Camera Calibration // Microsoft Research Tech. Rep. MS-98-71. – 1998. – 21 p.
4. Shi J., Tomasi C. Good Features to Track // Proc. IEEE Conf. on Computer Vision and Pattern Recognition. – Seattle, WA, USA, 1994. – P. 593–600.
5. Lucas B.D., Kanade T. An Iterative Image Registration Technique with an Application to Stereo Vision // International Joint Conf. on Artificial Intelligence. – 1981. – P. 674–679.
6. Tomasi C., Kanade T. Detection and Tracking of Point Features // Carnegie Mellon Univ., Tech. Rep. CMU-CS-91-132. – 1991. – 20 p.
7. Jin H., Favaro P., Soatto S. Real-time tracking and outlier rejection with changes in illumination // Proc. IEEE Intern. Conf. on Computer Vision. – 2001. – P. 684–689.
8. Hartley R.I. Self-Calibration of Stationary Cameras // Intern. J. of Computer Vision. – 1997. – V. 22. – № 1. – P. 5–23.
9. Методы компьютерной обработки изображений / под ред. В.А. Сойфера. – М.: Физматлит, 2003. – 784 с.
10. Van Aken J.R. An Efficient Ellipse Drawing Algorithm // Computer Graphics & Artificial Intelligence. – 1984. – V. 4. – № 9. – P. 24–35.

Поступила 03.04.2010 г.

УДК 519.688

АЛГОРИТМ РАСПАРАЛЛЕЛИВАНИЯ ОБРАБОТКИ ЦИФРОВОГО ВИДЕОПОТОКА

A.M. Кох

Томский университет систем управления и радиоэлектроники
E-mail: alexander.koh@maiconcept.ru

Предложен новый алгоритм декодирования цифрового видеопотока, в котором использовано распараллеливание вычислений в сочетании с синхронизацией реконструкции опорных и предсказанных блоков видеоинформации. Алгоритм в два раза производительнее существующих аналогов для видеопотоков стандартного разрешения и на треть для видеопотоков высокой четкости.

Ключевые слова:
Цифровое видео, параллельные алгоритмы, компенсация движения, синхронизация, MPEG, VC-1.

Key words:
Digital video, parallel algorithms, motion compensation, synchronization, MPEG, VC-1.

Семейство MPEG-форматов, состоящее из MPEG-2, MPEG-4, H.264 (AVC) и VC-1, является наиболее распространенным, в связи с чем задача разработки модификаций, направленных на повышение производительности алгоритмов сжатия и декодирования, остается актуальной.

С увеличением разрешения и битрейта цифрового видеопотока (до 1920×1080 с битрейтом до 40 Мбит/с для видеопотока высокого разрешения) применение алгоритмов распараллеливания для обработки цифровой видеоинформации выходит на передний план, позволяя использовать все возможности и преимущества многопроцессорной техники.

Известно, что изображения в MPEG-последовательности подразделяются на следующие типы [1]:

- *I* (*intra*), играющие роль опорных при восстановлении остальных изображений по их разностям;
- *P* (*predicted*), содержащие разность текущего изображения с предыдущим *I* или *predicted* с учетом смещений отдельных фрагментов;
- *B* (*bidirectionally predicted*), содержащие разность текущего изображения с предыдущим и последующим изображениями типов *I* или *P* с учетом смещений отдельных фрагментов.

Отдельные изображения состоят из макроблоков. Макроблок – основная структурная единица фрагментации изображения, он соответствует участку изображения размером 16×16 пикселей. Для них определяются векторы движения относительно *I*- или *P*-изображений. В свою очередь каждый макроблок состоит из нескольких блоков, количество которых определяется форматом сжатия, несущих информацию о яркости *Y*, цветовых *U*- и *V*-компонентах. Блоки являются базовыми структурными единицами, над которыми осуществляются основные операции кодирования, в том числе выполняется дискретное косинусное преобразование (DCT – Discrete Cosine Transform) и квантование полученных коэффициентов [2]. Векторы движения определяют расстояние между двумя фрагментами на экране, основываясь на количестве пикселей между этими областями.