УДК 004.93

Н.С. СВИРНЕВСКИЙ
Хмельницкий национальный университет

АЛГОРИТМ ТРЕХМЕРНОЙ РЕКОНСТРУКЦИИ С ОПРЕДЕЛЕНИЕМ ПОЛОЖЕНИЯ КАМЕРЫ ПО ОСОБЫМ ТОЧКАМ

В статье описывается алгоритм 3D-реконструкции по изображениям с определением положения камеры по особым точкам, который есть решением одной из множества задач, реализуемых через композицию матриц аффинных преобразований в однородных координатах. Алгоритм описан для простейшего случая ортогонального проецирования, когда взаимное положение камер определяется только 2-я параметрами. Однако этот алгоритм может быть легко обобщен для случаев, когда взаимное положение камер определяется большим количеством параметров, включая и центральное проецирование. Особые точки изображений выбираются и сопоставляются по алгоритму, универсальному для всех разновидностей точек.

Ключевые слова: 3D-реконструкция, особые точки, матрица, преобразование, алгоритм.

N.S. SVIRNEVSKYI
Khmelnytskyi National University

ALGORITHM FOR THREE-DIMENSIONAL RECONSTRUCTION WITH DETERMINATION OF POSITION OF THE CHAMBER ON SPECIAL POINTS

This article describes an algorithm for 3D-reconstruction of the images with the determination of the camera position on special points, which is the decision of one of the many problems to be implemented through the composition of the matrix of affine transformation in homogeneous coordinates. The algorithm is described in the simplest case of an orthogonal projection, when the relative position of the cameras is determined by only 2 parameters. However, the algorithm can be easily generalized to cases where the relative position of the cameras is determined by a large number of options, including a central projection. Specific points of images are selected and compared according to an algorithm that is universal for all varieties of points.

Keywords: 3D-reconstruction, special points, matrix, transformation, algorithm.

Введение

Трехмерная реконструкция – это процесс получения 3D объектов на основе изображений. На вход алгоритма обработки подается набор из нескольких изображений (два или более), результатом работы которого является 3D фотография (рис. 1). Поскольку изображения получены из разных точек наблюдения, то для получения соответствия между точками изображения необходимо определить (знать) параметры аппарата проецирования. Эти параметры могут быть определены через решение системы уравнений [1] по известному положению особых точек изображений. Особой точкой изображения называется точка изображения, окрестность которой существенно отличается от окрестности любой другой точки изображения.

Рис. 1. Система 3D видеонаблюдения

Анализ исследований и публикаций

Сопоставление изображений является одним из фундаментальных аспектов многих задач компьютерного зрения, таких как распознавание объектов, построение трёхмерной сцены на основе нескольких изображений, создание стереопары, создание панорамных изображений, motion tracking (слежение за объектом в видеопоследовательности) и так далее.

В 1981 году с выходом работы Моравеца [2] начались исследования сопоставления изображений с помощью нахождения особых точек. На протяжении уже более чем 30 лет было разработано множество методов решения этой задачи [2–5]. По причине того, что область знаний, решающая подобные проблемы, ещё молода, не существует определённого универсального метода, идеально подходящего для всех задач такого рода. Однако, существуют методы решения более узких задач, каждый из которых будет показывать лучшие результаты в зависимости от использования для конкретной задачи.

В предыдущих публикациях [1, 6, 7] автора этой статьи на основе достаточно наглядного и простого практического примера (сканирование рупорной антенны) демонстрируется механизм аффинных преобразований в пространстве. Рассматриваются прямая задача (построение изображений рупорной антенны) и обратная – восстановление параметров рупорной антенны по одному ее изображению [7] и 2-м

изображениям [1]. В статье [1] на основе аффинных преобразований описывается алгоритм решения задачи 3D-реконструкции объекта по 2-м изображениям для случая, когда положение камеры известно. Такая задача находит применение, например, при панорамной съемке, где камеры фиксированы (см. рис. 1).

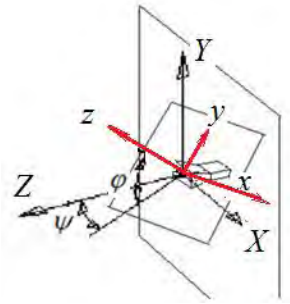
Формулирование цели

При съемках объекта фотографом с разных позиций положение камеры обычно не фиксируется. Как определить положение камеры непосредственно по изображениям? Для этого необходимо определить координаты характерных соответствующих точек, чтобы решить систему уравнений аффинных преобразований для пересчета координат в параметры положения камеры.

Ставится задача дополнить алгоритм, изложенный в статье [1], для случая, когда положение камеры неизвестно. Основная часть этой задачи заключается в разработке алгоритма детектирования особых точек изображений и определения соответствующих точек на изображениях

Изложение основного материала

Суть математического решения задачи восстановления пространственного объекта по его изображениям заключается в описании проекций точек объекта в различных системах координат (СК) и в установлении взаимосвязи между этими СК (рис. 2). При проецировании пространственного объекта на плоскость теряется одна размерность (координата Z). В статье [1] была определена система уравнений (1), которые устанавливают взаимосвязь между точками изображений и позволяют восстановить координату Z, если известно направление проецирования (2 угла, которые определяют положение камеры).



$$x = X\cos\psi - Z\sin\psi$$
$$y = X\sin\psi\sin\varphi + Y\cos\varphi + Z\cos\psi\sin\varphi \quad \textbf{(1)}$$
$$z = 0$$

Рис. 2. Установление взаимосвязи между точками объекта на 2-х изображениях

Систему уравнений (1) можно использовать для нахождения координаты Z каждой точки на главном изображении – точки с координатами (X,Y). Решение возможно, если известны параметры преобразования (значения углов) и координаты соответствующей точки (x,y) на вспомогательном изображении. Простейший алгоритм решения этой задачи:

1. Для каждой точки главного изображения (X,Y) определяем соответствующую точку на вспомогательном изображении (x,y), осуществляя итерацию координаты Z
2. Соответствующая точка найдена, если совпадают 3 цвета точек (RGB) на обоих изображениях.
3. Вместе с соответствующей точкой определяется и координата Z.

Для тестирования алгоритма восстановления 3D-объекта по изображениям была разработана программа [1], которая обеспечивает загрузку файлов изображений (рис. 3) и обработка точек изображений для определения координаты Z (рис. 4).

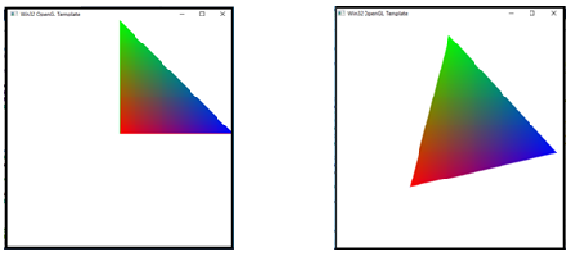


Рис. 3. Загрузка файлов изображений

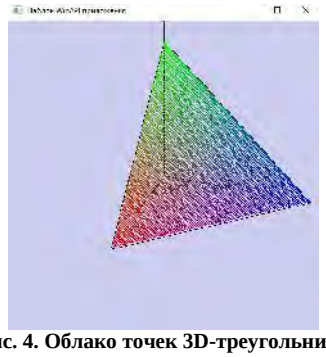


Рис. 4. Облако точек 3D-треугольника

Как определить положение камеры непосредственно по изображениям? Для этого необходимо решить систему уравнений для пересчета координат соответствующих точек одного изображения в координаты другого относительно параметров положения. При нашей упрощенной постановке задачи параметрами положения являются 2 угла.

$$x_1 = X_1\cos\psi - Z_1\sin\psi$$
$$y_1 = X_1\sin\psi\sin\varphi + Y_1\cos\varphi + Z_1\cos\psi\sin\varphi \quad (2)$$

В этой системе уравнений кроме углов неизвестной является еще координата Z. Следовательно, для решения задачи уравнений недостаточно. Запишем систему уравнений для еще одной пары соответствующих точек:

$$x_2 = X_2 \cos\psi - Z_2 \sin\psi$$
$$y_2 = X_2 \sin\psi \sin\varphi + Y_2 \cos\varphi + Z_2 \cos\psi \sin\varphi \quad (3)$$

Получили 4 уравнения с 4-я неизвестными. Система решается, если известны координаты двух соответствующих пар точек изображений. Т.о., для определения положения камеры необходимо:

1. Осуществить поиск особых точек на каждом изображении.
2. Установить соответствие между особыми точками изображений.
3. Определить параметры положения камеры – 2 угла в системе уравнений(1).

Существуют множество алгоритмов определения особых точек, предназначенных для разных областей применения. В них были определены следующие требования, которым должны удовлетворять особые точки:

1. Отличимость: особая точка должна выделяться на фоне и быть уникальной в своей окрестности.
2. Инвариантность: выбор особых точек должен быть независимым от аффинных преобразований.
3. Стабильность: выбор особых точек должен быть устойчив к шуму и ошибкам.
4. Уникальность: помимо локального отличия, особая точка должна также обладать свойством глобальной уникальности для того, чтобы улучшить различимость повторяющихся паттернов.
5. Интерпретируемость: особые точки должны определяться таким образом, чтобы их можно было использовать для анализа соответствий и лучшей интерпретации изображения.

Ниже изложен алгоритм и фрагменты программы нахождения характерных особенностей особых точек. Для определения особых точек и установления соответствия между точками на разных изображениях необходимо предварительно обработать изображение. Обработка изображения заключается в следующем:

1. Цветное 3-байтное изображение преобразуется в 1-байтное изображение (с оттенками серого цвета).
2. Цвета пикселов усредняются (сглаживаются) через матричные фильтры обработки изображений.

В компьютерном представлении широко распространённая серая шкала использует на каждый пиксел изображения один байт (8 бит) информации. Такая шкала передаёт 256 оттенков (градаций) серого цвета, или яркости (значение 0 представляет чёрный цвет, а значение 255 – белый). В цветовых пространствах яркость Y' вычисляется по формуле

$$Y' = 0.299R + 0.587G + 0.114B.$$

Матричный фильтр обработки изображений – это матрица коэффициентов, которая «умножается» на значение пикселей изображения для получения требуемого результата. В методах фильтрации при оценке реального сигнала в некоторой точке кадра принимают во внимание некоторое множество (окрестность) соседних точек, воспользовавшись определенной похожестью сигнала в этих точках. Эффективное шумоподавление можно осуществить, если влияние пикселей на результат будет уменьшаться с увеличением расстояния от обрабатываемого. Этим свойством обладает гауссовский фильтр:

0,000789	0,006581	0,013347	0,006581	0,000789
0,006581	0,054901	0,111345	0,054901	0,006581
0,013347	0,111345	0,225821	0,111345	0,013347
0,006581	0,054901	0,111345	0,054901	0,006581
0,000789	0,006581	0,013347	0,006581	0,000789

Особые точки характеризуются направлением наибольшего увеличения яркости и величиной её изменения в этом направлении, которые вычисляются с помощью оператора Собеля. Оператор использует ядра 3x3, с которыми сворачивают исходное изображение для вычисления приближённых значений производных по горизонтали и по вертикали. Пусть A – это исходное изображение, а Gx и Gy – два изображения, на которых каждая точка содержит приближённые производные по x и по y. Они вычисляются следующим образом:

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

В каждой точке изображения приближённое значение величины градиента можно вычислить путём использования полученных приближенных значений производных. Используя эту информацию, мы можем также вычислить направление градиента:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \qquad \mathbf{\Theta} = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

Фрагменты программной реализации

1. Считывается растр изображения из BMP файла в массив *rgb*:

```
…
RGBQUAD **rgb = new RGBQUAD*[bmiHeader.biWidth];
        for (int i = 0; i < bmiHeader.biWidth; i++) {
                rgb[i] = new RGBQUAD[bmiHeader.biHeight];
        }
        int kr = (int)bmiHeader.biWidth * 3 % 4;
        if (kr != 0) { kr = 4 - kr; }
        for (int j = 0; j < bmiHeader.biHeight; j++) {
```

```
                for (int i = 0; i < bmiHeader.biWidth; i++) {
                        rgb[i][j].rgbBlue = getc(pFile);
                        rgb[i][j].rgbGreen = getc(pFile);
                        rgb[i][j].rgbRed = getc(pFile);
                }
                // пропускаем последние 1-3 байта в конце строки для обеспечения кратности 4
                for (int i = 0; i < kr; i++) {getc(pFile);}
        }
```

2. Цветное 3-байтное изображение преобразуется в 1-байтное изображение (с оттенками серого цвета). Точки изображения сохраняются в 4-м резервном свойстве переменной *rgb[i][j]*:

```
void Bmp::RgbToGray (RGBQUAD **rgb, int mx, int my) {
        int i, j;
        for (j = 0; j < my; j++)  {
                for (i = 0; i < mx; i++) {
rgb[i][j].rgbReserved = rgb[i][j].rgbRed*0.299 + rgb[i][j].rgbGreen*0.587 + rgb[i][j].rgbBlue *
0.114;}}
}
```

3. Цвета пикселов усредняются (сглаживаются) в соответствии с ближайшими пикселами рисунка матрицей 5*5. В резервный байт перезаписываются обновленные цвета точек:

```
void Bmp::Filter(RGBQUAD **rgb, int mx, int my){
        int i, j, n, m;
        RGBQUAD **rgbtmp = new RGBQUAD*[mx];
        double g[5][5];
g[0][0] = 0.000789; g[1][0] = 0.006581; g[2][0] = 0.013347; g[3][0] = 0.006581; g[4][0] =
0.000789;
g[0][1] = 0.006581; g[1][1] = 0.054901; g[2][1] = 0.111345; g[3][1] = 0.054901; g[4][1] =
0.006581;
g[0][2] = 0.013347; g[1][2] = 0.111345; g[2][2] = 0.225821; g[3][2] = 0.111345; g[4][2] =
0.013347;
g[0][3] = 0.006581; g[1][3] = 0.054901; g[2][3] = 0.111345; g[3][3] = 0.054901; g[4][3] =
0.006581;
g[0][4] = 0.000789; g[1][4] = 0.006581; g[2][4] = 0.013347; g[3][4] = 0.006581; g[4][4] =
0.000789;
        for (i = 0; i < mx; i++) {rgbtmp[i] = new RGBQUAD[my];}
        for (j = 2; j < my - 2; j++)  {
                for (i = 2; i < mx - 2; i++) {
                        rgbtmp[i][j].rgbReserved = 0;
                        for (m = 0; m < 5; m++)  {
                                for (n = 0; n < 5; n++) {
                                        rgbtmp[i][j].rgbReserved = rgbtmp[i][j].rgbReserved + rgb[i
- (2 - n)][j - (2 - m)].rgbReserved*g[n][m];}}
                }
        }
        for (j = 2; j < my - 2; j++)  {
                for (i = 2; i < mx - 2; i++) {
                        rgb[i][j].rgbReserved = rgbtmp[i][j].rgbReserved;}
        }
        delete rgbtmp;
}
```

4. Реализуется поиск градиентов характеристик точек с помощью оператора Собеля и осуществляется отбор особых точек по пороговым значениям этих характеристик:

```
void Bmp::Gradient(RGBQUAD **rgb, int mx, int my){
        int i, j, n, m, grdX, grdY, grdX1, grdY1, grdR, grdR1, grdC;
        double grdCos, angle;  // cos угла градиента
        double gx[3][3];  // фильтр Собеля для определения градиента вдоль x
        gx[0][0] = -1.0; gx[1][0] = 0.0; gx[2][0] = 1.0;
        gx[0][1] = -2.0; gx[1][1] = 0.0; gx[2][1] = 2.0;
        gx[0][2] = -1.0; gx[1][2] = 0.0; gx[2][2] = 1.0;
        double gy[3][3];  // фильтр Собеля для определения градиента вдоль y
        gy[0][0] = 1.0; gy[1][0] = 2.0; gy[2][0] = 1.0;
        gy[0][1] = 0.0; gy[1][1] = 0.0; gy[2][1] = 0.0;
        gy[0][2] = -1.0; gy[1][2] = -2.0; gy[2][2] = -1.0;
        RGBQUAD **rgbtmp = new RGBQUAD*[mx]; // Создаем массив для градиентов характеристик точек
        for (i = 0; i < mx; i++) {rgbtmp[i] = new RGBQUAD[my];}
        // первые производные
        for (j = 3; j < my - 3; j++)  {
                for (i = 3; i < mx - 3; i++) {
                        grdX = 0;
                        grdY = 0;
                        for (m = 0; m < 3; m++)  {
                                for (n = 0; n < 3; n++) {
        grdX = grdX + rgb[i - (1 - n)][j - (1 - m)].rgbReserved * gx[n][m];  // градиент вдоль x
        grdY = grdY + rgb[i - (1 - n)][j - (1 - m)].rgbReserved * gy[n][m];  // градиент вдоль y
                                }
                        }
                        grdR = sqrt(pow(grdX, 2) + pow(grdY, 2)); // градиент результирующий
                        if (grdR) {
                                grdCos = (double) grdX / grdR;  // косинус угла градиента
            angle = (grdY >= 0) ? acos(grdCos) : 6.28 - acos(grdCos); // угол градиента
// Нормализуем (значения должны быть в пределах 0-255) и запоминаем характеристики пикселя
```

```
                        rgbtmp[i][j].rgbRed = grdR / 6; // запоминаем градиент текущего пиксела
                        rgbtmp[i][j].rgbGreen = angle * 180/6.28;
// запоминаем угол градиента текущего пикселя (1/2 угла с учетом ограничения 0-255)
                        }
                        else
                        {
                        rgbtmp[i][j].rgbRed = 0; // запоминаем градиент текущего пиксела
                        rgbtmp[i][j].rgbGreen = 0; // запоминаем угол градиента текущего пикселя
                        }
                }
        }
        // вторые производные
        for (j = 4; j < my - 4; j++)  {
                for (i = 4; i < mx - 4; i++) {
                        grdX = 0; grdY = 0; grdX1 = 0;grdY1 = 0;grdC = 0;
                        for (m = 0; m < 3; m++)  {
                                for (n = 0; n < 3; n++) {
                        grdX = grdX + rgbtmp[i - (1 - n)][j - (1 - m)].rgbRed * gx[n][m];
// изменение величины градиента  вдоль x
                        grdY = grdY + rgbtmp[i - (1 - n)][j - (1 - m)].rgbRed * gy[n][m];
// изменение величины градиента  вдоль y
                 // Через фильтр Собеля определяем изменение направления угла градиента
                        grdX1 = grdX1 + rgbtmp[i - (1 - n)][j - (1 - m)].rgbGreen * gx[n][m];
// изменение направления угла градиента вдоль x
                        grdY1 + rgbtmp[i - (1 - n)][j - (1 - m)].rgbGreen * gy[n][m];
// изменение направления угла градиента вдоль y
                                        continue;
                                }
                        }
grdR = sqrt(pow(grdX, 2) + pow(grdY, 2)); // изменение величины градиента
grdR1 = sqrt(pow(grdX1, 2) + pow(grdY1, 2)); // изменение направления угла градиента
// Нормализуем (значения должны быть в пределах 0-255) и запоминаем характеристики пикселя
rgbtmp[i][j].rgbBlue = grdR; // запоминаем изменение величины градиента
rgbtmp[i][j].rgbReserved = grdR1/4; // запоминаем изменение направления угла градиента
                                continue;
                }
        }
        int angl, za, it, jt, r=10;
        // Проверка условий
        for (j = r; j < my - r; j++)  {
                for (i = r; i < mx - r; i++) {
        angl = 2*rgbtmp[i][j].rgbGreen;
//угол градиента текущего пикселя(1 / 2 угла с учетом ограничения 0 - 255)
                        if (angl < 23)za = 1;                           //  0 (360) градусов
                        if (angl > 23  && angl <= 67 )za = 2;       //  45 градусов
                        if (angl > 67  && angl <= 102)za = 3;       //  90 градусов
                        if (angl > 102 && angl <= 157)za = 3;       //  135 градусов
                        if (angl > 157 && angl <= 202)za = 5;       //  180 градусов
                        if (angl > 202 && angl <= 247)za = 6;        //  225 градусов
                        if (angl > 247 && angl <= 292)za = 7;       //  270 градусов
                        if (angl > 292 && angl <= 337)za = 8;       //  315 градусов
                        if (angl > 337)za = 1;                          //  0 (360) градусов
                        switch (za){
                        case 1:  it = i + r;  jt = j; break;  //  0 (360) градусов
                        case 2:   it = i + r;  jt = j + r; break; //  45 градусов
                        case 3:   it = i; jt = j + r; break;       //  90 градусов
                        case 4:   it = i - r; jt = j + r;  break;  //  135 градусов
                        case 5:   it = i - r; jt = j;  break;      //  180 градусов
                        case 6:   it = i - r; jt = j - r; break;   //  225 градусов
                        case 7:   it = i;  jt = j - r;   break;    //  270 градусов
                        case 8:   it = i + r;  jt = j - r; break;   //  315 градусов
                        }
        int prgrd = 10; // порог по градиенту
        int prgrd1 = 30; // порог по изменению величины градиента (точка перегиба)
        int prgrd2 = 20; // порог по сравнению градиентов на расстоянии r
        int prangle = 15; // порог по изменению направления угла градиента

rgb[i][j].rgbReserved = (rgbtmp[i][j].rgbRed >= prgrd
                        && rgbtmp[i][j].rgbBlue <= prgrd1
                        && rgbtmp[i][j].rgbRed - rgbtmp[it][jt].rgbRed >= prgrd2
                        && rgbtmp[i][j].rgbReserved >= prangle
                                        )
                                        ? 1 : 0;
                }
        }
}
```

5. Все точки изображения выводятся на экран, особые точки выделяются на изображении по цвету и размерам (рис. 5).

```
                                clr     =     RGB(rgb[i][j].rgbRed,     rgb[i][j].rgbGreen,
rgb[i][j].rgbBlue);
                                SetPixel(hdc, p.x, p.y, clr);

                                if (rgb[i][j].rgbReserved == 1){
                                        clr = RGB(0, 0, 0);
```

```
                //clr = RGB(255, 255, 255);
                //SetPixel(hdc, p.x, p.y, clr);
                 Ellipse(hdc, p.x - 4, p.y - 4, p.x + 4, p.y + 4);
}
```

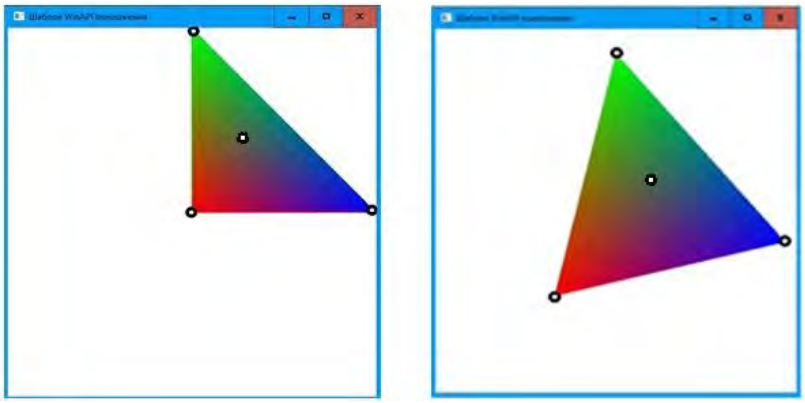


Рис. 5. Выделение программой особых точек изображений

Кроме выделения особых точек на изображениях необходимо установить соответствие между проекциями одной и той же точки на различных изображениях. Для однозначного решения этой задачи одних лишь количественных градиентных характеристик яркости точек может быть недостаточно. Необходимо определять вид каждой точки. Так различают точки пятна и угловые точки. В зависимости от количества пересекаемых граней существуют разные виды уголков: L-, Y- (или T-) и X-связные (рис. 6).

Рис. 6. Разновидности угловых точек

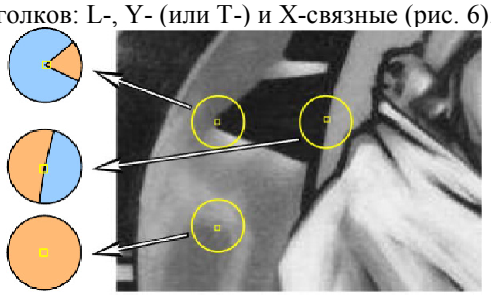


Рис. 7. Особые точки – контурные и угловые

Среди отфильтрованных по градиентам особых точек могут быть те, которые расположены на границе контуров, и угловые (рис. 7). Угловые точки являются наиболее характерными. Углы определяются сегментацией круговых окрестностей в схожие (оранжевые) и непохожие (синие) участки.

Особенность угловой точки должно подчеркивать все, что может:

1. Наиболее интенсивный переход цвета от одной к другой области.
2. Области, которые граничат, должны быть равномерными по цвету (или одинаковыми по текстуре) и большими по площади.
3. Угловая точка должна характеризоваться резким изменением угла касательной, перед точкой и далее линия должна изменяться монотонно.
4. Примыкающие к точке линии должны быть достаточно длинными.

Особенность точки-пятна:

1. Небольшая замкнутая область с интенсивным переходом цвета от одной области к другой.
2. Форма пятна имеет правильную форму (окружность, эллипс, квадрат).
3. Точка находится по центру фигуры.

Воспользуемся так называемым алгоритмом SUSAN. Углы находятся там, где относительная площадь схожих участков (similar USAN) достигает локального минимума (ниже определенного порога). Отдалимся от переходного участка изменения интенсивности цвета, создав вокруг тестируемой особой точки окружность радиуса 3-4 пиксела. Определяем абсолютные значения разностей интенсивностей цвета точек на противоположных концах диаметрального отрезка. Если разность близка к нулю (ниже порогового значения), то точки находятся на одном участке. Если разность большая (выше порогового значения), то точки находятся на разных участках. Пошагово изменяем положение диаметрального отрезка, вращая его на угол до 180 градусов. Если разность для всех положений близка к нулю (ниже порогового значения), то особая точка является светлым (темным) пятном. Если количество положений с большой и маленькой разностями интенсивностей в диаметрально противоположных точках одинакова, то особая точка является вершиной прямого угла, при других отношениях количества точек углы меньше или больше (до 180 градусов). Если разность для всех положений большая (выше порогового значения), то особая точка является либо точкой границы контура, либо вершиной, в которой сходятся 3-и и более участков различного цвета. Для последнего случая нужен дополнительный анализ.

На основании вышеизложенного предлагается следующий алгоритм выбора точки-угла и точки пятна. Вокруг тестируемой точки определяем интенсивности точек на концентричных окружностях 2-х радиусов (чтобы не ловились случайные одиночные точки необходимо сравнивать не с интенсивностью

тестируемой точки, а с интенсивностью точек на окружности меньшего радиуса). Предполагается, что точки окружности меньшего (большего) радиуса находятся в пределах (за пределами) внутреннего участка. Сканируем точки обеих окружностей с определенным угловым шагом. Разность интенсивностей тестируемой точки и точек на меньшей (большей) окружности должна быть стабильно низкой (высокой). Если для всех сканируемых точек выполняются эти условия, то тестируемая точка находится в области пятна. Окружности можно заменить 8-ю направлениями (4-е основные направления и 4-е диагональные). Например, изменение интенсивности точек внутреннего (внешнего) участка определяется матрицей 3*3 (матрицей 5*5).

По каждому из 8 направлений вокруг точки сравниваем величину градиентных характеристик (в простейшем случае яркости) точки dR;

Если dR> Pmax, тогда kmax++

Если dR< Pmin, тогда kmin++.

Порог Pmax определяет, что интенсивность точек вне угла резко изменяется. Порог Pmin определяет, что в пределах сегмента угла точки однородны – имеют одинаковую интенсивность.

При kmax=kmin=4 – угол 180 градусов, т.е. точка является контурной, но не угловой.

При kmax>4 и kmin=8 – kmax точка является вершиной угла меньше 180 градусов.

При kmax=8 точка является пятном. Пятно может быть особой точкой в случае, если область вокруг нее является однородной. Поэтому точки вокруг пятна необходимо проверить на однородность (dR< Pmin). При этом все точки по 7 направлениях сравниваются с точкой по первому направлению.

Выводы

Рассмотренный в статье алгоритм решения задачи трехмерной реконструкции с определением положения камеры по особым точкам прост и доступен для понимания, поскольку представлен как решение одной из множества задач, реализуемых через композицию матриц аффинных преобразований в однородных координатах. Особые точки изображений выбираются и сопоставляются по алгоритму, универсальному для всех разновидностей точек.

В статье алгоритм описан для простейшего случая ортогонального проецирования, когда взаимное положение камер определяется только 2-я параметрами. Однако этот алгоритм может быть легко обобщен для случаев, когда взаимное положение камер определяется большим количеством параметров, включая и центральное проецирование.

Литература

1. Свирневский Н.С. Алгоритм реконструкции трехмерной модели по изображениям / Н. С. Свирневский // Вісник Хмельницького національного університету. Технічні науки. – 2017. – № 2. – С. 212–218.

2. Moravec H. Rover visual obstacle avoidance // In International Joint Conference on Artificial Intelligence, Vancouver, Canada. – 1981. – P. 785–790.

3. Жук Д.В. Восстановление трёхмерной модели сцены по цифровым изображениям / Д.В. Жук, А.В. Тузиков, А.В. Бородач // Искусственный интеллект. – 2006. – № 2. – С. 142–146.

4. Аленин В. А. Трехмерная реконструкция объектов из последовательности изображений / В.А. Аленин // Молодой ученый. – 2011. – № 3. Т. 1. – С. 33–36.

5. Борисенко Д.И. Методы поиска угловых особенностей на изображениях / Д.И. Борисенко // Молодой ученый. – 2011. – № 5. Т. 1. – С. 120–123.

6. Свирневский Н.С. Основы разработки графических приложений / Н.С. Свирневский, С.С. Ковальчук. – Хмельницкий : ХНУ, 2015. – 270 с.

7. Свирневский Н.С. Восстановление параметров пространственного объекта по изображению / Н.С. Свирневский // Вісник Хмельницького національного університету. Технічні науки. – 2011. – № 6. – С. 74–78.

References

1. Svyrnevskyi N.S. Alhorytm rekonstruktsyy trekhmernoi modely po yzobrazhenyiam / N. S. Svyrnevskyi // Herald of Khmelnytsky National University. – 2017. – # 2. – S. 212–218.

2. Moravec H. Rover visual obstacle avoidance // In International Joint Conference on Artificial Intelligence, Vancouver, Canada. – 1981. – P. 785–790.

3. Zhuk D.V. Vosstanovlenye trekhmernoi modely stseny po tsyfrovym yzobrazhenyiam / D.V. Zhuk, A.V. Tuzykov, A.V. Borodach // Yskusstvennyi yntellekt. – 2006. – # 2. – S. 142–146.

4. Alenyn V. A. Trekhmernaia rekonstruktsyia obektov yz posledovatelnosty yzobrazhenyi / V.A. Alenyn // Molodoi uchenyi. – 2011. – # 3. T. 1. – S. 33–36.

5. Borysenko D.Y. Metody poyska uhlovykh osobennostei na yzobrazhenyiakh / D.Y. Borysenko // Molodoi uchenyi. – 2011. – # 5. T. 1. – S. 120–123.

6. Svyrnevskyi N.S. Osnovy razrabotky hrafycheskykh prylozhenyi / N.S. Svyrnevskyi, S.S. Kovalchuk. – Khmelnytskyi : KhNU, 2015. – 270 s.

7. Svyrnevskyi N.S. Vosstanovlenye parametrov prostranstvennoho obekta po yzobrazhenyiu / N.S. Svyrnevskyi // Herald of Khmelnytsky National University. – 2011. – # 6. – S. 74–78.

Рецензія/Peer review : 14.02.2017 р. Надрукована/Printed :24.10.2017 р.
Рецензент: д.т.н., проф. Сорокатый Р.В.