

文章编号:1001-9081(2005)02-0329-03

## 基于显微镜图像的三维场景重建技术研究

陆芸婷,陈有青,李振军

(中山大学信息科学与技术学院,广东 广州 510275)

(Lu-yunting@163.com)

**摘要:**描述了一种基于显微镜图像下的二维融合图的三维重建技术,该技术可以还原出二维融合图像的三维特征。在分析原理的基础上,使用 Direct3D 在 VC 环境下实现了三维重建技术。

**关键词:**融合;三维重建;显微镜图像;Direct3D

**中图分类号:** TP391.41 **文献标识码:** A

## Study on 3D-scene rebuilding technology based on microphone

LU Yun-ting, CHEN You-qing, LI Zhen-jun

(College of Information Science &amp; Technology, Sun-YatSen University, Guangzhou Guangdong 510275, China)

**Abstract:** In this paper, a technology of 3D rebuilding based on microphone's graphs was presented, which could revert 3D characters of 2D-fixed graphs. Depended on theory, 3D rebuilding could be realize by Direct3D in VC compiling environment.

**Key words:** unite; 3D rebuild; microphone; Direct3D

## 0 引言

随着时代的发展,显微镜越来越多的应用到各领域。人们对信息的获取已由传统的二维显微图像变为三维立体图形。那么,如何把显微镜下的二维图形以它尽可能真实的三维体凸显在计算机屏幕上就是我们要研究的问题。

本文以微软的 DirectX 中的 Direct3D 为开发工具,实现显微图像的三维场景重建工作。

物体三维重建是指通过一定方法获取物体表面上一系列点在某一参考坐标系的三维坐标及表面纹理信息,然后通过立体视觉理论在计算机上重现其立体模型。它在生物医学、工业制造、建筑工程等领域有广泛的应用。

显微镜的三维图像要求有明显的层次性。本文是以多幅图像的二维融合图像为基础,还原出二维融合图像的三维特征。融合图的信息取自每幅图中最清晰的部分,三维模型要求融合图的三维信息来自不同图的部分高度不同,层次分明,使得它具有明显的三维高度,强烈的层次性。

目前它已广泛的应用于刑侦,文检工作中,以对指纹,血迹,齿痕,弹道及文书等做出正确的解析。

## 1 基础理论

## 1.1 基于纹理特征的多幅图像的二维融合

由于显微镜视野的有限性,我们观察物体时得到的二维图像总是某一部分清晰,而其他部分是模糊的。把多幅这些局部清晰的图融合在一起取每幅图最清晰的部分就能得到一幅完全清晰的二维图像了。这个过程是二维图像的融合过程。我们采用基于纹理特征的图像融合方法来实现。

纹理是表征像素按一定的排列规律重复排列的参数。纹

理分析方法大致可分为:统计方法、构造方法和图谱分析方法。我们采用的是统计方法,它适用于无法判断结构要素和规则的图像的分析,最常用的是以像素的灰度和空间位置为参数的分析方法,也就是基于灰度共生矩阵计算纹理特征的方法。

空间灰度层共生矩阵方法,是建立在估计图像的二阶组合条件概率密度函数基础上的。

假定待分析的纹理图像为一矩形图像,水平和垂直方向上各有  $N_c$  和  $N_r$  个像元,将每个像元上出现的灰度量化为  $N_q$  层。设  $Z_c = \{1, 2, \dots, N_c\}$  为水平空间域,  $Z_r = \{1, 2, \dots, N_r\}$  为垂直空间域,  $G = \{1, 2, \dots, N_q\}$  为量化灰度层集。集  $Z_r \times Z_c$  为行列编序的图像像元集,则图像函数  $f$  可以表示为一个函数:指定每一个像元具有  $N_q$  个灰度层中的一个值  $G$ ,即:

$$f: Z_r \times Z_c \rightarrow G$$

灰度共生矩阵是描述在  $\theta$  方向上,相隔  $d$  像元距离的一对像元,分别具有灰度层  $i$  和  $j$  的出现概率,其元素可记为  $P(i, j | d, \theta)$ ,当  $\theta$  和  $d$  选定时,也可记为  $P_{i,j}$ 。显然灰度层共生矩阵是一个对称矩阵,其阶数由图像中灰度层数决定。

下面分别给出  $0^\circ$ 、 $45^\circ$ 、 $90^\circ$  和  $135^\circ$  方向上的灰度共生矩阵的数学描述:

$$P_{0^\circ, d}(a, b) = |\{(k, l), (m, n)\} \in D: k - m = 0, |l - n| = d, f(k, l) = a, f(m, n) = b| |$$

$$P_{45^\circ, d}(a, b) = |\{(k, l), (m, n)\} \in D: (k - m = -d, l - n = d) \text{ OR } (k - m = d, l - n = -d), f(k, l) = a, f(m, n) = b| |$$

$$P_{90^\circ, d}(a, b) = |\{(k, l), (m, n)\} \in D: |k - m| = d, l - n = 0, f(k, l) = a, f(m, n) = b| |$$

$$P_{135^\circ, d}(a, b) = |\{(k, l), (m, n)\} \in D: (k - m = d,$$

收稿日期:2004-07-29

作者简介:陆芸婷(1980-),女,安徽六安人,博士研究生,主要研究方向:数据库与电子商务; 陈有青(1950-),女,广西柳州人,副教授,主要研究方向:计算机美术、电子商务; 李振军(1979-),男,广西柳州人,硕士研究生,主要研究方向:网络与工程软件。

$l - n = d$ ) OR  $(k - m = -d, l - n = -d), f(k, l) = a, f(m, n) = b$  | |

其中:  $D = (Nr \times Nc) \times (Nr \times Nc)$

由灰度层共现矩阵可以计算出一组参数,用来描述纹理特性,4种比较常用的参数为:

$$1) \text{ 能量: } E(d, \theta) = \sum \{p(i, j | d, \theta)\}^2$$

$$2) \text{ 熵: } H(d, \theta) = \sum \{p(i, j | d, \theta)\} \times \log \{p(i, j | d, \theta)\}$$

$$3) \text{ 惯性矩: } I(d, \theta) = \sum (i - j)^2 p(i, j | d, \theta)$$

$$4) \text{ 相关: } C(d, \theta) = \sum (i - u_x)(j - u_y)p(i, j | d, \theta) / \sigma_x \sigma_y$$

其中:

$$u_x = \sum i \sum \{p(i, j | d, \theta)\}$$

$$u_y = \sum j \sum \{p(i, j | d, \theta)\}$$

$$\sigma_x = \sum (i - u_x)^2 \sum \{p(i, j | d, \theta)\}$$

$$\sigma_y = \sum j \sum (j - u_y)^2 \{p(i, j | d, \theta)\}$$

这四个参量分别反应纹理一致性、纹理的熵、纹理对比和纹理相关性。

以反应图像纹理对比的惯性矩作为比较参量进行比较提取。通过一行和列的扫描,对每个像素点,计算它为左上角的  $16 \times 16$  矩阵的纹理值,然后对各幅图进行比较,选取最大者表示该点属于清晰部分,即可得到我们想要提取的清晰部分。

## 1.2 三维数据提取

三维数据来自二维图像的融合。根据二维图像的融合过程,把要融合的二维图像从1开始依次编号。并设立一图像大小的二维数组。计算每个像素点的纹理值,取最大者那幅图的像素作为融合图的像素。同时将该图编号记入到数组中相应位置。融合过程完成,二维数组也置满。此二维数组作为我们三维重构的数据信息。数组中的值即为重建过程该像素对应的三维信息中的高度值。

其过程用图1来表示。其中黑点表示像素点。

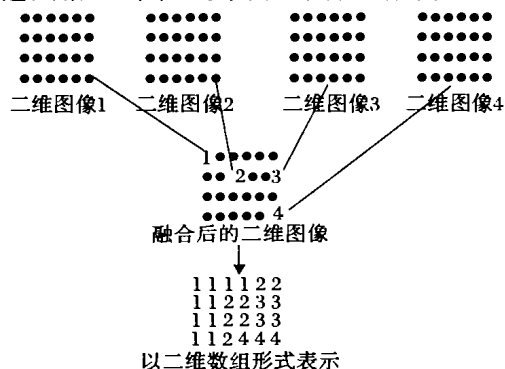


图1 二维图像的融合过程

## 2 三维重建

### 2.1 基本步骤

先对这些二维图像进行融合、存储,得到三维数据场。再进行三维曲面的构图,经简单平滑处理,得到一个粗略三维图像。再用 Directx 对它进行建模处理,生成三维模型。最后对三维模型进行贴图操作,把二维融合图像贴在三维模型上。

至此,三维重构完成。高质量的二维图像和合适的三维重建方法提高了本研究中的三维图像的真实性和精确性。

### 2.2 三维造型

三维造型的第一步是定义物体的空间坐标。它代表物体的真实情况,源于真实世界的独立于程序和显示器。第二步,把顶点变换到摄像机空间,以获得一个合适的视点。摄像机在原点向正  $z$  方向看(Direct3d 使用左手坐标系,所以  $z$  的正方向指向场景)。把世界坐标围绕着一个摄像机的位置和方向重新定位得到的新的坐标叫做观察坐标。第三步,利用投影公式在显示屏上显示图像所得的投影坐标。投影坐标是用户最后在屏幕上看到的坐标。它是一个二元组。

三维图元就是形成单个三维实体的顶点集合。Direct3D 用三角形组成大多数多边形,因为一个三角形中的3个顶点肯定是共面的。可以通过组合多个三角形来形成大而复杂的多边形和网格。也可以使用三角形创建表面呈现平滑曲线的图元。使用 Direct3D 进行三维模型的构造时,要先对各点的坐标进行定义。它以一中心点对其余各点画三角形,先以  $v1, v2, v3$  三点画三角形,再以  $v1, v3, v4$  为顶点画三角形,依次类推,见图2。坐标定义顺序是把二维数组按如下格式划分,每9个点围成一正方形。先对中心点定义,然后再从左上角开始按顺时针方向定义一周各十个点。每个正方形均如此。这样组成的三角图元顺序为  $(1, 2, 3), (1, 3, 4), (1, 4, 5) \dots (1, 9, 10)$ , 见图3。



图2 三维模型显示过程

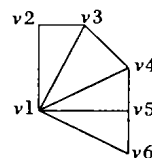


图3 三角图元

构造完三维模型只是建三维场景的第一步,要想有较好的效果还必须确定物体的表面材质和纹理。

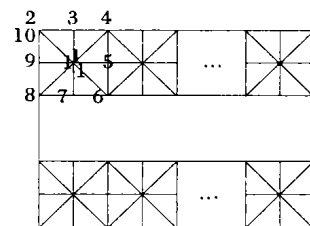


图4 二维数组的点的定义顺序

### 2.3 三维场景生成

#### 2.3.1 系统结构

我们对该系统的软件设计采用了面向对象的设计方法。主要包括二维图像融合、二维图像处理、三维重建和场景数据库模块。二维图像融合主要是将显微镜下的多幅二维图像融合成一幅清晰图像并在屏幕上显示出来。同时把三维数据以二维数组的形式传给三维重建模块。三维重建模块完成二维显微图像的立体化,以突出的层次性显示在屏幕上。

#### 2.3.2 Direct3D

Direct3D 提供了整套用于处理三维场景的对象库及众多能够直接操纵计算机硬件的 API 函数让编程者调用。它分为保留模式和立即模式。保留模式是一组高层 3D API 函数,适用于 3D 程序的快速开发。立即模式由少量相关的,创建对象的 API 元素组成,它更适用于开发游戏程序和其他高性能

的多媒体程序。Direct3D 应用程序不管采用保留模式还是立即模式,都用一种类似的模式与图形硬件通信。

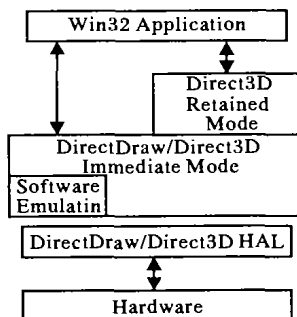


图5 立即模式

### 2.3.3 过程

本系统采用Direct9的立即模式基于VC进行编程设计。

1) 创建窗口对象;在执行任何Windows应用程序时首先必须创建一个显示用户界面的应用程序窗口。

2) 创建D3D对象;使用Direct3D9类的方法创建Direct3D渲染设备。一个应用程序通过调用IDirect3D9::CreateDevice()方法创建一个渲染设备。设置表达参数,最后创建Direct3D设备。

```
if ( FAILED( g_pD3D -> CreateDevice(
    D3DADAPTER_DEFAULT, D3DDEVTYPE_HAL, hWnd,
    D3DCREATE_SOFTWARE_VERTEXPROCESSING,
    &d3dpp, &g_pd3dDevice)) )
{
    return E_FAIL;
}
```

3) 定义坐标矩阵:设置空间坐标,观察坐标及投影坐标。

4) 初始化场景:设置用于实时渲染用的材质、光源及部分贴图。初始化几何体,调用IDirect3D9::CreateVertexBuffer()方法创建一个顶点缓存,并填充该缓存。Direct3D应用程序为其顶点使用D3DVERTEX结构,D3DVERTEX结构的成员描述顶点的位置和方向。用一个顶点法线向量指定方向。法线向量直指视口,它是世界坐标系的原点。下面给出正方形中中心点的定义:

```
g_Vertices[index].x = i;
g_Vertices[index].y = m_array[i][j];
g_Vertices[index].z = j;
g_Vertices[index].nx = 0.0f;           //法线向量的x分量
g_Vertices[index].ny = 0.0f;           //法线向量的y分量
g_Vertices[index].nz = 1.0f;           //指向原点后面的法线
g_Vertices[index].tv = ((float)i)/49;
//仅当使用纹理时使用,定义纹理坐标
g_Vertices[index].tu = ((float)j)/49;
//仅当使用纹理时使用
```

5) 监控系统消息:在创建了应用程序窗口之后,创建Directx对象,然后初始化场景,这就为渲染场景做好了准备。在大多数情况下,Windows应用程序在其消息循环中监控系统消息,然后,每当队列中没有消息时就渲染帧。

```
MSG msg;
ZeroMemory( &msg, sizeof( msg ) );
while( msg.message != WM_QUIT )
{
    if ( PeekMessage( &msg, NULL, 0U, 0U,
        PM_REMOVE ) )
    {
        TranslateMessage( &msg );
```

```
        DispatchMessage( &msg );
    }
    else
        Render();
}
```

6) 渲染并显示场景:调用IDirect3DDevice9::Clear方法清除视口。清除以后,该实例通知Direct3D开始渲染,接着渲染场景,然后发出已经渲染完成的信号。Direct3D使用Drawprimitive的方法会构造出由D3DVERTEX类型的三角形组成的立体几何体。如下所示:

```
g_pd3dDevice -> Clear(0, NULL, D3DCLEAR_TARGET,
    D3DCOLOR_XRGB(255,255,255), 1.0f, 0);
// Begin the scene
if ( SUCCEEDED( g_pd3dDevice -> BeginScene() ) )
{
    .....
    for ( int i = 0; i < iSquareCount; i++ )
    {
        g_pd3dDevice -> DrawPrimitive(
            D3DPT_TRIANGLEFAN, i*10, 8);
    }
    // End the scene
    g_pd3dDevice -> EndScene();
}
```

7) 关闭:在执行中间的某些点时,应用程序必须关闭。关闭一个DirectX应用程序不仅意味着要销毁应用程序窗口,而且还必须对应用程序所使用的DirectX对象解除分配并使其指针失效。

```
if ( g_pVB != NULL )
    g_pVB -> Release();
if ( g_pd3dDevice != NULL )
    g_pd3dDevice -> Release();
if ( g_pD3D != NULL )
    g_pD3D -> Release();
```

8) 为了使生成的几何体能达到更好的效果,还要进行材质、光照和贴图等操作以使三维模型更加逼真。在这里用于贴图的图片是前面二维融合后得到的新的融合图像。

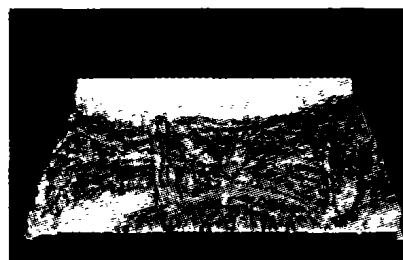


图6 融合图像

## 3 结语

本文描述了基于VC环境下Direct3D实现基于显微镜图像的三维场景重建过程。采用这种方法后我们可以更加直观地感受到显微图像的三维特征。

### 参考文献:

- [1] 姜渊胜,朱跃龙,黄河. 基于虚拟现实技术的实时漫游系统研究及实现[J]. 计算机工程,2003,(27):98-99.
- [2] 武永康. Direct3D原理与API参考[M]. 北京:清华大学出版社,2001.
- [3] 杨玲,陈文家,赵明扬. 在VC环境下用Direct3D IM Framework开发三维动画[J]. 计算机应用,2002,(22):106-108.