中文摘要

动态序列图像分析是计算机视觉研究领域中十分热门的问题，它的基本任务是从图像序列确定物体三维运动参数及结构参数。基本分析方法有两种：基于特征的方法和基于光流的方法。由于经典光流法对噪声敏感，算法费时，不利于对目标进行实时跟踪。因此本文结合两种方法，采用稀疏光流法进行计算。

首先，本文采用微软公司提供的 VFW (Video For Windows) 软件包开发纯软件的视频采集程序，提高了程序的通用性；

其次，对采集的图像进行图像平滑、阈值分割和边缘跟踪等一系列预处理，以去处图像噪声，分离目标物体和背景，并提取出目标物体的边缘点，为稀疏光流场的计算做好准备；

再次，采用面积约束和夹角约束相配合的方法快速准确地提取特征点，并利用具有强大并行处理能力的 Hopfield 神经网络来完成序列图像间特征点的松弛匹配，在此基础上利用视差计算特征点处的稀疏光流；

最后，利用稀疏光流进行三维物体运动参数和结构参数的估计。采用散焦法估计物体深度，并进行深度修正以满足测量精度要求，从而解决了单目视觉估计深度的难题；在获取物体深度信息的条件下，采用一种从光流场恢复物体三维运动参数的强壮算法，只需要 5 个特征点就可以唯一确定物体三维运动参数。

关键词： 序列图像分析，稀疏光流，特征点，散焦法，单目视觉

ABSTRACT

The analysis of dynamic image sequences is a quite hot topic in computer vision, and its basic task is to estimate 3-D motion and structure parameters from a sequence of images. There are two basic analysis approaches: the feature-based approach and the optical flow based approach. Because the classical optical flow algorithm is sensitive to noise and time-consuming, it is not extremely suitable for tracking timely. Thus, we adopt sparse optical flow approach by combing that two approaches in this paper.

Firstly, in this paper, we develop video capture program completely by software method with VFW software package provided by Microsoft Corporation, and in this way the generality of the program is improved.

Secondly, in order to eliminate image noise, separate the object from background and extract edge points of the object, a series of pre-processings are carried out to the images captured, such as image smoothing, threshold segmenting and edge tracking. This makes preparations for the computation of sparse optical flow field.

Thirdly, we extract feature points of the object quickly and exactly by the method of combining area constraint and angle constraint, and use the Hopfield neural network with massively parallel processing capability to perform relaxation matching of feature points within sequences of images. On this basis we compute sparse optical flow of the feature points by means of parallax.

Finally, we estimate 3-D motion and structure parameters of the body by sparse optical flow. In order to solve the monocular vision problem of estimating depth estimation, we estimate the depth of the object by means of position-from-defocus, and carry out depth adjusting to meet the measure precision demand. Under the condition of acquiring depth information, 3-D motion parameters of the object can be estimated uniquely only using 5 feature points with a robust algorithm of recovering 3-D motion parameters of the object from optical flow field.

Key words: image sequences analysis, sparse optical flow, feature point, position-from-defocus, monocular vision

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 天津大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：[signature]　　签字日期：2004 年 1 月 2 日

学位论文版权使用授权书

本学位论文作者完全了解 天津大学 有关保留、使用学位论文的规定。特授权 天津大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：[signature]　　导师签名：王刚

签字日期：2004 年 1 月 2日　　签字日期：2004 年 1月 2 日

第一章 绪论

1.1 选题背景及意义

动态图像是针对运动的物体或景物而言的，他们不仅是空间位置的函数，而且是随时间变化的，它为我们提供了比单一图像更丰富的信息。在对某一景物拍摄到的图像序列中，相邻两帧图像间至少有一部分像元的灰度发生了变化，这个图像序列就称之为动态图像序列[1]。

动态图像分析的基本任务是从图像序列中检测出运动信息，识别与跟踪运动目标和估计运动目标三维运动及结构参数。它涉及到图像处理、图像分析、人工智能与模式识别、计算机视觉等研究领域，是图像处理与计算机视觉领域中的一个非常活跃的分支，在工业生产、医疗卫生、国防建设等领域得到了广泛应用，因此对它的研究具有十分重要的现实意义。

早期的序列图像处理主要是针对电视信号的处理和传输，如基于三维运动的图像分割、增强、滤波、编码压缩等，故仍属于图像处理的范畴。随着计算机视觉研究的深入，三维运动及结构信息的处理已成为序列图像处理的主题，同时研究的难度也随之增大。1979 年在美国费城召开了第一次序列图像处理的专题讨论会，以后关于该领域的会议、专刊及著作日益增多，可以说它已成为当今计算机视觉的一大热点。国内开展动态序列图像分析研究最早是在 80 年代初，北方交通大学信息所在此取得了国内领先的成果。以后相继一些大学和科研机构也在这一领域做了不少工作，如复旦大学、浙江大学、北京大学等，目前已形成了全国范围的精干强大的科研队伍，有关动态序列图像分析的研究日新月异，不断有新的成果出现。

目前动态序列图像分析主要有两类方法：一类是基于光流的连续处理方法，一类是基于特征的离散处理方法。

1.1.1 基于特征的方法

基于特征的方法利用了特征位置的变化信息，通常分为三步：一，从相继两幅或多幅不同时刻的图像中抽取显著特征，如与拐角、边界、有明显标记的区域对应的点、线或曲线等。二，在图像帧间寻找特征点的对应关系，也称匹配。三，依据这些特征之间的对应来计算物体的结构(形状、位置等)和运动信息。

Ullman[2]指出，由三幅正投影图像中四组对应点可确定刚体三维运动参数和

不共面四点结构的精确模型，Zhuang[3]和 Huang[4]给出了解决该问题的线性算法。但正投影只适用于一些特殊情况，在真实世界中一般都采用透视投影。

Roach 和 Aggarwal[5]通过透视投影变换从视图中计算物体的结构和运动。他们认为相继两个视图上的五个对应点对恢复运动参数和结构参数是必要的。由于位置变量有 27 个，而方程只有 20 个，因此他们假设第一幅视图对应的摄像机 6 个参数为 0 和绝对深度为任意值。所以这种方法不能得到绝对深度。

Tsai[6]和 Longuet-Higgins[7]考虑了曲面物体的情况，分别独立提出了一种利用两幅图像中 8 对对应点计算刚体旋转和平移矩阵的算法，他们都十分重视计算的唯一性。Longuet-Higgins[8]后来指出，退化的结构会影响他们的 8 点算法。Zhuang[9]给出了一种通用的线性算法，高[10]建立了两种运动分解方式下的刚体运动方程，给出了一种求解旋转矩阵的线性算法，证明一般情况下有两组解。

Yen 和 Huang[11]提出了迭代解法，用球面投影方法提出了三个视图七条线之间的对应，他们把刚体运动分解为绕通过原点轴线的转动和一个移动。对于纯旋转运动，两条线在两个视图之间的对应对决定旋转矩阵已经足够充分。对于纯移动，通过三个视图五条线之间的对应可产生一个线性方程系统，据此可以求解平移运动。对于一般情况，可用三个视图六条线之间的对应求解。但是该方法对噪声比较敏感。

基于特征方法的优点在于抗噪性能好，适用于长时间大运动量的处理。但复杂点是特征提取和对应问题，并且目前的方法在求解三维运动参数中还含有一个需要先验知识才能确定的因子。

1.1.2 基于光流的方法

光流的概念是 Gibson 于 1950 年首先提出的。所谓光流是指图象中模式运动的速度。光流场是一种二维(2D)瞬时速度场，其中的 2D 速度矢量是景物中可见点的三维(3D)速度矢量在成象表面的投影。光流不仅包含了被观察物体的运动信息，而且携带着有关景物三维结构的丰富信息。

光流计算首先是由 Horn 和 Schunck[12]提出的。1981 年，Horn 和 Schunck 提出了计算光流的基本等式，但利用基本等式来求解光流场是一个不适定问题，这是因为每一点上的光流都是一个速度向量，它有两个分量，而由运动引起的图像上每一点亮度的变化仅仅提供了一个约束，因此必须引入一些附加的约束条件，才能得到唯一解。

Horn 等人依据同一运动物体所产生的光流场是连续和平滑的假设，提出了

在光流场上附加约束（整体平滑约束），将光流计算问题转化为一个变分问题，用来解决光流计算的不适定问题。Lucas 和 Kanade[13]提出了局部光滑的约束条件，即假设在一个小的空间邻域Ω上，景物的运动矢量保持恒定，然后使用加权最小二乘法估计光流。Terzopoulos[14]和 Hildreth[15]也分别提出了更好的有向平滑度约束条件。Yachida[16]在此基础上提出了迭代算法，使光滑性约束条件不仅涉及空间域也涉及时间域。

Haralick 等[17]的光流计算方法是将三维物体的表面分割成许多小的平面，他们假设三维物体上每个小平面有近似的运动参数且运动参数在短时间内为常量，依据此思想提出了计算光流附加约束。Tretiak 等[18]提出了不同的思路，他们将速度看作局部常量，直接对基本等式求偏导，得到了与 Haralick 等的光流计算方法相似的算法。Nagel[19]全面地对灰度场附加约束进行了研究。国内复旦大学吴立德教授领导的课题组曾对灰度时变图象的光流场计算理论和方法进行过研究，并提出了多通道方法[20]。

除了基于灰度的方法以外，计算光流的方法还有基于能量的方法[21]、基于区域的光流场计算方法[22]、基于像素递归的光流场计算方法、基于随机平滑度约束条件的贝叶斯光流场分析法，以及傅立叶自适应平滑约束方法和几何约束方法等。Beauchemin 等[23]对光流的计算做了一个很好的全面综述。

虽然基于光流的方法不需要进行连续图像间特征的匹配，但存在着某些缺点。首先，光流的计算需要微分运算，而图象的微分运算是噪声敏感的；其次光流的计算常用松弛法的迭代运算，算法费时，难以满足实时控制的要求。

1.2 研究方法的选择

1.2.1 单目视觉方法的选择

计算机视觉系统获取物体三维信息最简单的方法是立体视觉方法，即用两台摄像机对同一物体摄取图像，根据两幅图像的视差来得到物体的三维信息。该方法仿照人的双眼获取物体的三维信息，原理简单，但该方法有一个难以解决的匹配问题。为了计算三维物体某一点的空间数据，必须在两台摄像机的像平面上找出对应于同一物体点的两个像点，当要计算的物体点很多时，这个找对应匹配像点的问题是很难解决的。一般采用相关算子方法，而对应点的搜索沿内极线进行。

采用单目视觉，即用一台摄像机摄取动态序列图像来进行物体三维运动和结

构参数的估计，就可以避免多摄像机的匹配问题，比较容易应用于实际。但是用单目视觉估计物体深度信息是极为困难的，一般认为，由于从像点出发通过焦点所作的射线上的无数个点都与这个像点对应，这就造成了物体绝对深度信息的丢失。

Pentland[24]于 1987 年提出了用聚焦图像和散焦图像恢复物体绝对深度的算法。在摄像机标定的情况下，根据散焦图像扩展函数的σ值，就可以方便的得到物点的深度。这是用单目摄像机且只用一幅图像恢复物体深度有成效的方法。如果通过调节透镜组焦距或图像平面与透镜之间的距离，得到聚焦图像，则得到的物体深度极为准确。Subbarao[25]推导了阶跃型边缘散焦图像恢复物体深度的算法，并提出了线扩展函数的概念。他的方法涉及到对图像的微分运算，而微分运算是噪声敏感的，因而只能得到深度的不精确估计。Shang-Hong Lai[26]提出了由模糊边缘恢复物体深度的通用算法，此方法去边缘附近点的灰度值来估计物体深度。他们的算法是对目标函数进行优化，迭代算法计算时间很长（长达 24 秒）难于用在实时跟踪中。

本文利用散焦法估计物体深度，并进行深度修正以满足测量精度的要求，从而解决了单目视觉估计深度的难题。在已知深度的条件下，对序列图像进行运动分析，再从而恢复物体的三维运动参数。

1.2.2 稀疏光流方法的选择

由于基于特征的方法和基于光流的方法各有其优缺点：基于光流的方法只依赖于图像灰度值的变化，不需要进行图像间的匹配，但对噪声敏感，仅适用于短时间小运动量的处理，且计算速度慢；基于特征的方法对噪声不太敏感，适用于长时间大运动量的处理，且计算速度快，但图像间的特征匹配比较困难。

为了扬长避短，本论文汲取两种方法的优点，采用了一种混合法----稀疏光流法，即首先利用基于特征的方法在一幅图像上抽取相对稀疏、但有明显差别的角点，并在相继两幅图像间进行特征点的匹配；其次利用视差计算出特征点的光流，并利用稀疏光流进行物体三维运动参数和结构参数的估计。

采用稀疏光流法有以下优点：

（1）对噪声不敏感　因为计算的仅是特征点的光流，只要预先对图像进行必要的预处理，使特征点的提取较准确，就能减小噪声的影响；并且计算光流时没有用到微分运算，也可大大减少噪声的影响。

（2）运算速度快　利用基于光流的方法需要计算图像上每一个像素处

的光流（稠密光流），而该方法仅计算特征点的光流，所以速度要快得多；

（3）可以计算帧间大的视差运动，边缘不被模糊，并且比较适合多目标的场合。

近年来有不少文章提到稀疏光流（sparse flow）的方法[27]，但由于提取特征点并在序列图像间进行特征点匹配比较困难，有的文章采用人为指定特征点的方法来获取特征点，有的方法仅适用于物体表面纹理结构特别强的情况，这些都无法满足实际需要。本论文在提取特征点前对图像进行了一系列的预处理，在提取特征点时又采用了面积约束和夹角约束相配合的方法，能够准确快速的提取特征点，采用 Hopfield 神经网络[28]实现序列图像间特征点的松弛匹配，从而很好地解决了匹配问题。

1.3 本论文的主要研究内容

本论文首先采用单目视觉方法获取实时视频图像，并对采集的视频图像进行特定的预处理，去除图像的噪声，加强图像中有用信息；然后利用本文提出的稀疏光流法计算特征点处的光流场；最后利用特征点处的稀疏光流进行三维重构，估计目标物体的三维运动参数和结构参数。

本论文共分六章：

第一章为绪论，主要简要介绍序列图像分析的研究现状及意义、本文研究方法的选择和主要研究内容。

第二章为视频图像采集与预处理，主要介绍用 VFW 软件包进行视频图像的实时采集，并对采集的视频图像进行图像增强、阈值分割等预处理，以及如何显示和存储图像；

第三章介绍稀疏光流场的计算过程，主要包括：首先利用面积约束和夹角约束相配合的方法进行特征点的提取，其次采用 Hopfield 神经网络实现序列图像间特征点的松弛匹配，最后利用视差计算特征点处的稀疏光流；

第四章为由稀疏光流场来恢复物体的三维结构信息和运动信息，首先利用散焦法估计物体的深度，然后在已知深度信息的条件下，利用 5 个特征点处的光流来恢复物体的三维运动参数。

第五章为程序设计及实验，在对程序进行介绍的基础上，做了大量实验来验证本文所述方法的正确性。

第六章为结论和展望。

第二章 视频图像采集与预处理

视频图像采集是进行序列图像分析的前提，本文采用 VC++ 提供的 VFW (Video For Windows) 函数库开发纯软件的视频采集程序。由于外界环境和输入设备的影响，直接从摄像机采集到的图像质量较差，含有大量噪声，图像不清晰，不利于提取特征点和进行特征点匹配，因此要对采集的初始图像进行一系列的预处理。

2.1 视频图像的实时采集

用 Microsoft 公司推出的 VFW 软件可以对视频图像进行采集、编辑和播放等操作，但要把视频采集功能添加到自己的应用程序当中，必须自己编写程序。实现方法有两种：第一种方法是利用视频设备生产厂商提供的软件开发工具箱（SDK）来开发应用程序；第二种方法是使用 VC++提供的 VFW 函数库，开发基于 Windows 的视频捕获应用程序。用前者编程具有很大的局限性：它依赖于硬件，即采集卡改变，编写的图像采集程序也随之改变，大大降低了工作效率；本文采用VFW函数库编写纯软件的、适合任何视频采集卡的通用视频采集程序。

2.1.1 AVICap 窗口类简介

运用 VFW 函数库进行视频采集编程主要是通过 AVICap 窗口类来实现，它是一个简单易用的、基于消息的接口，用户可以用它来访问视频及音频采集硬件，并控制采集视频流到硬盘的过程。AVICap 窗口类的主要功能为：

- 进行音频、视频采集，并将采集到的数据保存到一个.avi 文件中。也可以指定保存数据的文件名，并可将该文件复制到另外一个文件；
- 动态地将视频和音频输入设备与应用程序连接和断开；
- 以预览（Preview）或叠加（Overlay）模式显示输入的实时视频信号；
- 可以设置视频采集的帧率；
- 可以显示控制视频源和视频格式的对话框；
- 可以创建、保存和装载与视频图像相关的调色板，并可将它复制到剪切板；
- 可以将采集到的单帧图像保存为 DIB(Device Independent Bitmap)格式。

2.1.2 用 AVICap 窗口类实现视频采集

为了实现视频采集，AVICap 窗口类提供了大量的宏、消息和函数，这里只按照采集程序的一般过程介绍几个关键的宏和函数的使用方法，其他宏和函数的使用方法可以参阅 VC++6.0 的 MSDN 帮助。

1. 创建采集窗口

采集窗口是被采集的视频图像的预览窗口，它是所有采集操作的基础。采集窗口是通过调用 capCreateCaptureWindow 函数创建的，如果函数调用成功，则返回捕获窗口的句柄。

2. 将采集窗口和指定的采集设备相连

我们可以用宏 capDriverConnect(gWndCap，nIndex)将采集窗口和指定的采集设备相连接。其中 gWndCap 为采集窗口的句柄，nIndex 为指定的采集设备的号码，该索引号允许的范围是 0-9。当有多个视频设备时，捕获窗口可以根据索引号选择指定的视频设备进行连接。我们可以通过 capGetDriverDescription 获得系统中所有采集设备的名称和版本号。

调用宏 capDriverDisconnect(gWndCap)可将采集窗口和指定的采集设备断开。

3. 获得与采集窗口相连的硬件驱动的能力

CAPDRIVERCAPS 结构中定义了采集驱动器的能力，如有无视频叠加能力，有无控制视频源和视频格式的对话框等。我们定义了 CAPDRIVERCAPS gCapDriverCaps，通过调用宏 capDriverGetCaps(gWndCap, &gCapDriverCaps, sizeof(CAPDRIVERCAPS))，在 gCapDriverCaps 中就复制了与当前采集窗口相连的驱动设备的各项能力，这时可以通过对话框完成对视频采集的控制。

4. 设置捕获窗口的显示模式

显示视频有预览和叠加两种模式。用宏 capPreview 将显示模式设置成 Preview 模式，而用宏 capOverlay 则将显示模式设置成 Overlay 模式。Preview 模式把数字化的帧从采集硬件传到系统内存，并利用 GDI 函数在采集窗口中显示，占用大量 CPU 时间，大部分设备都支持预览模式；而 overlay 模式将采集缓冲区的内容直接在监视器上显示，不用 CPU 资源，但只有一部分视频卡支持叠加模式，可调用宏 capDriverGetCaps 来判断视频设备是否具备 Overlay 模式。

用预览模式显示：

```
capPreviewRate(gWndCap, 66);     //设置显示帧率
capPreview(gWndCap, TRUE);     //开始预览显示
```

要结束预览显示可用：

```
capPreview(gWndCap, FALSE);
```

叠加模式与此方法相似，只是把 capPreview 改为 capOverlay。

本文采用预览模式显示视频，因为在后面的视频采集中，我们要通过回调函数得到内存缓冲区中视频数据的地址，因此所采用的显示方式必须具有把内存作为缓冲区来存放视频数据的能力，它是获得视频数据的必要条件。

5. 获取和设置采集窗口参数

使用宏 capCaptureGetSetup 可以获取当前采集参数设置，该宏得到的参数放在 CAPTUREPARMS 结构中，这个结构包含的参数控制视频流捕获过程，如捕获速率、捕获时使用的缓冲量、捕获如何终止等；可以对 CAPTUREPARMS 结构中的参数进行修改，然后调用宏 capCaptureSetSetup 进行更新。

6. 注册回调函数

回调函数是针对 Windows 中的窗口而设置的，回调过程由系统完成，而函数的具体内容则由程序员设定。系统中某一回调函数被设定后，在某一特定的条件满足时，系统自动调用该回调函数。AVICap 窗口类共提供了七种回调函数，常用的主要有以下四种回调函数：

1) 预览回调函数 capSetCallbackOnFrame( )

用它登记的回调函数在每采集完一帧后被调用。利用这个回调函数可以通过 VIDEOHDR 结构中的 lpData 指针得到所采集数据并对它进行处理。

2) 数据流回调函数 capSetCallbackOnVideoStream( )

用它登记的回调函数在采集的视频流图像存盘之前被调用；

3) 状态回调函数 capSetCallbackOnStatus( )

用它登记的回调函数在采集窗口发生改变时被调用；

4) 出错回调函数 capSetCallbackOnErrors( )

用它登记的回调函数在采集过程出错时被调用，通过编写相应的回调函数代码我们可以及时对错误进行处理。

7. 采集图像到缓存或文件并进行相应处理

视频采集主要有两种形式：单帧采集和连续视频流采集。

1) 单帧采集

我们可以调用宏 capGrabFrame (gWndCap)和 capGrabFrameNoStop(gWnd-Cap)来进行单帧采集。前者会产生图像冻结效果，而后者从视频采集卡获得的帧数据不被压缩地存入视频缓冲区中，之后将其显示出来，对预览或叠加模式的视频显示没有影响。

对于获得的单帧图像我们可以进行三种处理：调用宏 capFileSaveDIB( )把缓冲区的图像转化成 DIB 位图保存到磁盘、调用宏 capEditCopy(gWndCap)拷贝到剪切板和放在帧缓冲区以便及时进行处理。

2) 连续视频流采集

连续视频流采集也有两种方法：

```
capCaptureSequence(gWndCap);            //可把采集到的视频流数据存盘
capCaptureSequenceNoFile(gWndCap);  //不保存采集到的视频流数据
```

和单帧采集一样，视频流采集既可以将视频流采集到 AVI 文件中进行存盘，也可以利用登记的回调函数，把视频流数据先放在帧缓冲区，再对帧缓冲区内的视频数据进行实时处理。

2.2 数字图像预处理

2.2.1 灰度直方图

灰度直方图是数字图像处理中一个最简单、最有用的工具，它描述了一幅图像灰度级的内容。任何一幅图像的直方图都包括了可观的信息，某些类型的图像还可以由其直方图完全描述。

灰度直方图是灰度值的函数，描述的是图像中具有该灰度值的像素的个数，其横坐标表示像素的灰度级别，纵坐标是该灰度出现的概率（像素的个数）。绘制处理前后两幅图像的灰度直方图，可以直观地对处理前后的图像进行比较。

2.2.2 图像增强

图像增强的方法分为空域法和频域法两类，空域法主要是对图像中的各个像素点进行操作，它可以用下式来描述：

$$g(x,y) = f(x,y) \cdot h(x,y) \tag{2-1}$$

其中 $f(x,y)$是处理前的图像；$g(x,y)$表示处理后的图像；$h(x,y)$为空间运算函数。

频域法是在图像的某个变换域内（通常是频率域中）对图像的变换值进行操作，然后变换回空间域。例如：可以先对图像进行傅立叶变换，再对图像的频谱进行某种修正（如滤波等），最后将修正后的图像进行傅立叶反变换到空间域中，从而增强该图像，它是一种间接处理方法，可以用图 2-1 来描述该过程。

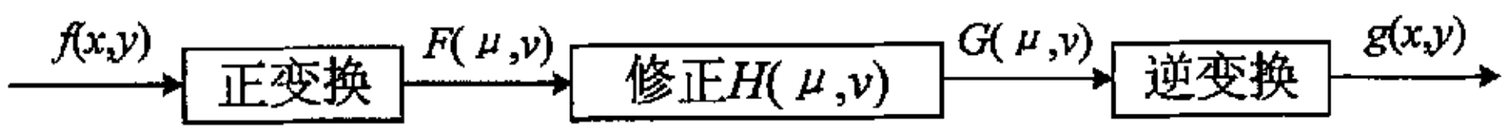


图 2-1 频率域增强模型

1. 图像平滑

图像平滑主要是为了消除噪声，它包括空域法和频域法两大类：在空间域内可以用领域平均来减少噪声；在频率域，由于噪声频谱通常多在高频段，因此可以采用各种形式的低通滤波方法来减少噪声。在空域法中，图像平滑的常用方法是采用中值滤波或均值滤波，对于中值滤波，就是指把以某点(x,y)为中心的小窗口内的所有像素的灰度按从大到小的顺序排列，将中间值作为(x,y)处的灰度值(若窗口中有偶数个像素，则取两个中间值的平均)。但对于细节多，特别是点、线、顶点多的图像不宜采用中值滤波；对于均值滤波，常采用模板法来实现，即通过一个点和它周围的几个点的某种运算（通常是平均运算）来消除突然变化的点，从而滤掉一定的噪声。由于本课题提取的特征点为六角螺栓的顶点，所以应该采用均值滤波进行图像平滑。

均值滤波常用的模板有：

平均模板(Box 模板)：
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1^{*} & 1\\ 1 & 1 & 1\end{bmatrix} \tag{2-2}$$

高斯模板(Gauss 模板)：
$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4^{*} & 2\\ 1 & 2 & 1\end{bmatrix} \tag{2-3}$$

中间的*表示中心元素，即用该元素做为处理后的元素。例如[2* 1]表示将自身的 2 倍加上右边的元素作为新值，而[2 1*]则表示将自身加上左边元素的 2 倍作为新值。

平滑模板的思想是通过一点和周围 8 个点的平均来去除突然变化的点，滤掉一定的噪声。它虽然考虑了邻域点的作用，但没考虑各点位置的影响，对于

所有的 9 个点都一视同仁，所以平滑的效果并不理想；而高斯模板考虑了离某点越近的点对该点的影响应该越大，引入了加权系数，使平滑的效果比较理想。

如下例所示：设图像为 $\begin{bmatrix}2 & 2 & 2 & 2\\3 & 2 & 3 & 3\\4 & 6 & 4 & 5\\5 & 6 & 6 & 6\end{bmatrix}$，分别用上述两种模板进行处理（周围元素直接从原图拷贝），结果为：

采用模板 1：$\begin{bmatrix}2 & 2 & 2 & 2\\3 & 3.11 & 3.22 & 3\\4 & 4.33 & 4.56 & 5\\5 & 6 & 6 & 6\end{bmatrix}$

采用模板 2：$\begin{bmatrix}2 & 2 & 2 & 2\\3 & 3 & 3.06 & 3\\4 & 4.56 & 4.56 & 5\\5 & 6 & 6 & 6\end{bmatrix}$

可以看到，原图中出现噪声的区域是第 2 行第 2 列和第 3 行第 2 列，灰度从 2 跳到 6，用 Box 模板处理后，灰度从 3.11 跳到 4.33；用高斯模板处理后，灰度从 3 跳到 4.56，都缓和了跳变的幅度，从这一点上看，两者都达到了平滑的目的。但是，原图中的第 3，第 4 行总的来说，灰度值是比较高的，经模板 1 处理后，第 3 行第 2 列元素的灰度变成了 4.33，与第 3、第 4 行的总体灰度相比偏小，而且原图中第 3 行第 2 列元素的灰度为 6，第 3 行第 3 列元素的灰度为 4，变换后，后者 4.56 反而比前者 4.33 大了。而采用高斯模板考虑了位置的影响，没有出现这些问题。

模板运算是一项非常耗时的运算。以式(2-3)为例，每个像素完成一次模板操作要用 9 次乘法，8 次加法和 1 次除法。对于一幅 N*N(宽度×高度)的图象，就要用到 $9(N-2)^2$ 次乘法，$8(N-2)^2$ 次加法和 $(N-2)^2$ 次除法，算法复杂度为 $O(N^2)$。为了提高运算速度，本文将 2 维模板运算转换成 1 维模板运算，例如式 (2-3)可以分解成一个水平模板和一个竖直模板，即：

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4^* & 2\\1 & 2 & 1\end{bmatrix}=\frac{1}{4}\begin{bmatrix}1\\2^*\\1\end{bmatrix}\times\frac{1}{4}\begin{bmatrix}1 & 2^* & 1\end{bmatrix}=\frac{1}{16}\begin{bmatrix}1\\2^*\\1\end{bmatrix}\times\begin{bmatrix}1 & 2^* & 1\end{bmatrix} \qquad (2\text{-}4)$$

这样，改进后将进行 $6(N-2)(N-1)$次乘法，$4(N-2)(N-1)$次加法和 $(N-2)^2$ 次除法操作，大大减少了乘法和加法运算。

例如：设图像为$\begin{bmatrix}1&2&4&2\\3&2&5&3\\4&6&4&5\\5&6&8&6\end{bmatrix}$，直接经过模板$\frac{1}{16}\begin{bmatrix}1&2&1\\2&4*&2\\1&2&1\end{bmatrix}$处理后变为

$\frac{1}{16}\begin{bmatrix}1&2&4&2\\3&53&61&3\\4&77&81&5\\5&6&8&6\end{bmatrix}$，采用分解后的模板来处理，结果为：

$$\frac{1}{16}\begin{bmatrix}1\\2*\\1\end{bmatrix}\times\begin{bmatrix}1&2*&1\end{bmatrix}\times\begin{bmatrix}1&2&4&2\\3&2&5&3\\4&6&4&5\\5&6&8&6\end{bmatrix}=\frac{1}{16}\begin{bmatrix}1\\2*\\1\end{bmatrix}\times\begin{bmatrix}1&9&12&2\\3&12&15&3\\4&20&19&5\\5&25&38&6\end{bmatrix}=\frac{1}{16}\begin{bmatrix}1&2&4&2\\3&53&61&3\\4&77&81&5\\5&6&8&6\end{bmatrix}\tag{2-5}$$

两者计算结果相同。

2. 梯度锐化

图像平滑往往使图像的边界、轮廓变的模糊，而图像锐化可以使图像的边缘、轮廓线以及图像的细节变的清晰。平滑处理后图像变模糊的根本原因是：图像受到了平均或积分运算。因此对图像进行逆运算（如微分运算）就可以使其变的清晰，常用的方法是梯度锐化。

设图像为$f(x,y)$，定义$f(x,y)$在点(x,y)处的梯度矢量$\vec{G}[f(x,y)]$为：

$$\vec{G}[f(x,y)]=\begin{bmatrix}\frac{\partial f}{\partial x}\\\frac{\partial f}{\partial y}\end{bmatrix}\tag{2-6}$$

梯度的幅度用$G[f(x,y)]$表示，其值为

$$G[f(x,y)]=\sqrt{\left(\frac{\partial f}{\partial x}\right)^2+\left(\frac{\partial f}{\partial y}\right)^2}\tag{2-7}$$

因为梯度的方向在函数$f(x,y)$最大变化率方向上，因此梯度的数值就是$f(x,y)$在其最大变化率方向上的单位距离上增加的量。

对于离散的数字图像，上式可以改写为：

$$G[f(i,j)]=\sqrt{[f(i,j)-f(i+1,j)]^2+[f(i,j)-f(i,j+1)]^2}\tag{2-8}$$

为了计算方便，也可以采用下面的近似计算公式：

$$G[f(i,j)] \cong |[f(i,j)-f(i+1,j)]| + |[f(i,j)-f(i,j+1)]| \tag{2-9}$$

$$G[f(i,j)] \cong |[f(i,j)-f(i+1,j+1)]| + |[f(i+1,j)-f(i,j+1)]| \tag{2-10}$$

式（2-9）和（2-10）的示意图如图 2-2 所示。

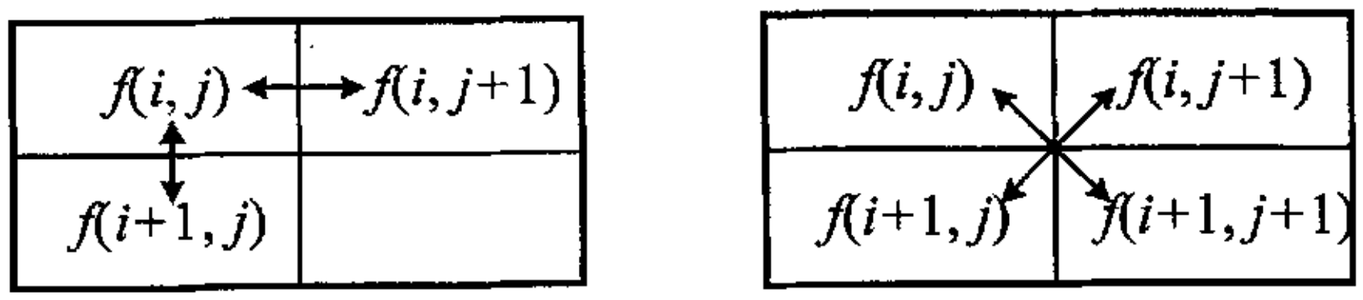


图 2-2 计算公式示意图

如果直接采用梯度值 $G[f(x,y)]$来表示图像，即令 $g(x,y)=G[f(x,y)]$，则由上面的公式可见：在图像变化缓慢的地方其值很小（对应于图像较暗）；而在线条轮廓等变化较快的地方其值很大。这就使得图像在经过梯度运算后变清晰，从而达到锐化的目的。

由于在图像变化缓慢的地方梯度很小，所以图像会变得很暗，我们采用以下方法来优化：先给一个阈值 Δ，如果梯度值 $G[f(x,y)]$小于该阈值 Δ，则保持原灰度值不变；否则赋值为 $G[f(x,y)]$：

$$g(x,y)=\begin{cases} G[f(x,y)] & (G[f(x,y)] \geq \Delta) \\ f(x,y) & (G[f(x,y)] < \Delta) \end{cases} \tag{2-11}$$

或者：

$$g(x,y)=\begin{cases} La & (G[f(x,y)] \geq \Delta) \\ f(x,y) & (G[f(x,y)] < \Delta) \end{cases} \tag{2-12}$$

其中 La 为一个固定的灰度值。通过适当地选取 Δ，可以有效地增强边界而不影响平滑区域的图像特征。

2.2.3 图像分割

图像分割是把图像空间分割成一些有意义的区域。阈值分割是图像分割的一种方法，进行阈值分割的目的是为了分割目标物体和背景。首先要确定一个阈值，对于序列图像采取动态调整阈值，然后让图像各像素点的灰度值与该阈值进行比较，如果大于该阈值，就把该像素点的灰度值置为 255（表示背景），

否则把该像素点的灰度值置为 0（物体），这样就把物体和背景区分开来了。经过阈值分割的图像就变成了二值图，只有 0 和 255 两种灰度值。

阈值的正确选取是取得良好分割效果的关键，本文采用一种迭代算法来求图像的最佳阈值。这一算法的步骤如下：

(1) 求出图像中的最小和最大灰度值 Z_l 和 Z_K，令阈值初值

$$T^0 = \frac{Z_l + Z_K}{2} \tag{2-13}$$

(2) 根据阈值 T^K 将图像分割成目标和背景两部分，求出两部分的平均灰度值 Z_O 和 Z_B；

$$Z_O = \frac{\sum_{z(i,j)<T^k} z(i,j) \times N(i,j)}{\sum_{z(i,j)<T^k} N(i,j)} \tag{2-14}$$

$$Z_B = \frac{\sum_{z(i,j)>T^k} z(i,j) \times N(i,j)}{\sum_{z(i,j)>T^k} N(i,j)} \tag{2-15}$$

式中 $Z(i,j)$ 是图像上 (i,j) 点的灰度值，$N(i,j)$ 是 (i,j) 点的权值系数，一般 $N(i,j)$ 取 1.0。

(3) 求出新的阈值；

$$T^{K+1} = \frac{Z_O + Z_B}{2} \tag{2-16}$$

(4) 如果 $T^K = T^{K+1}$，则结束，否则 $K \leftarrow K+1$，转步骤(2)。

根据上面的算法对图像进行阈值分割，效果比较理想。

2.2.4 轮廓跟踪与种子填充

轮廓跟踪的目的是为了获得图像的外部轮廓特征；种子填充是轮廓跟踪的逆过程。它们处理的都是经过阈值分割的二值图。

1. 轮廓跟踪

轮廓跟踪的基本方法是：先根据某些严格的“探测准则”找出目标物体轮廓上的像素，再根据这些像素的某些特征用一定的“跟踪准则”找出目标物体上的其它像素。

迄今，已有许多有关轮廓跟踪的算法，但大多数算法计算量比较大，速度

慢，不能满足实时跟踪的要求，本文采用一种快速算法，能够满足实际的需要。该算法的思路是：首先按照从左到右，从下到上的顺序搜索，找到的第一个黑点一定是最左下方的边界点；然后从该边界点开始，定义初始的搜索方向为沿左上方，如果左上方的点是黑点，则为边界点，否则搜索方向顺时针旋转 45 度，这样一直找到第一个黑点为止；最后把这个黑点作为新的边界点，在当前搜索方向的基础上逆时针旋转 90 度，继续用同样的方法搜索下一个黑点，直到返回最初的边界点为止。图 2-3 为这一轮廓跟踪算法的示意图，其中箭头代表搜索方向。

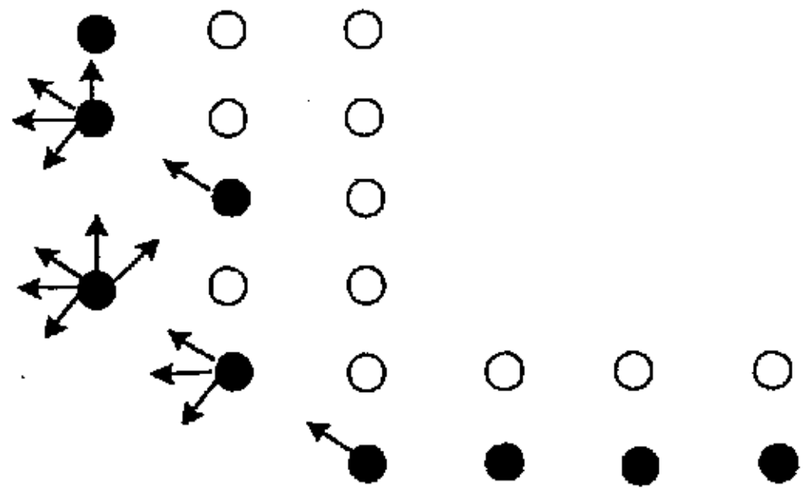


图 2-3 轮廓跟踪算法

2. 种子填充

本文采用种子填充是为了把边界内外点的像素值区分开来，边界外的点像素值为 255，而边界上和边界内的点经种子填充后都置为 0，为后面的特征点提取服务。

种子填充算法的突出优点是能对具有任意复杂边界的区域进行填充。然而，种子填充算法需要建立一个堆栈来存放种子，并有大量重复的种子进栈。当填充的区域面积较大时，由于所占用的栈空间太大，从而使填充不能完成、甚至会出现系统崩溃。解决上述问题的办法之一是扫描线种子填充算法。用这个算法，每一条连续未被填色的扫描线段只取一个种子进栈。从而栈空间大大减小。扫描线种子填充算法是一种能对任意复杂边界进行有效填充的实用算法。

我们称当前正在填色的连续扫描线段为当前线段；称当前线段上方的扫描线为上方扫描线；而在上方扫描线上各连续的未被填色的线段为上行线段；称当前线段下方的扫描线为下方扫描线；而在下方扫描线上各连续的未被填色的线段为下行线段。又称取自当前线段的种子为母种子，称取自各上、下行线段的种子为子种子，参见图 2-4。假定正在填充的扫描线是 y＝5，则线段 2<=x<=12 就是当前线段。扫描线 y=4 为上方扫描线，而线段 y＝4，4<=x<=7 和 y＝4，

11<=x<=13 为上行线段。扫描线 y=6 为下方扫描线，而线段 y=6，4<=x<=12 为下行线段。

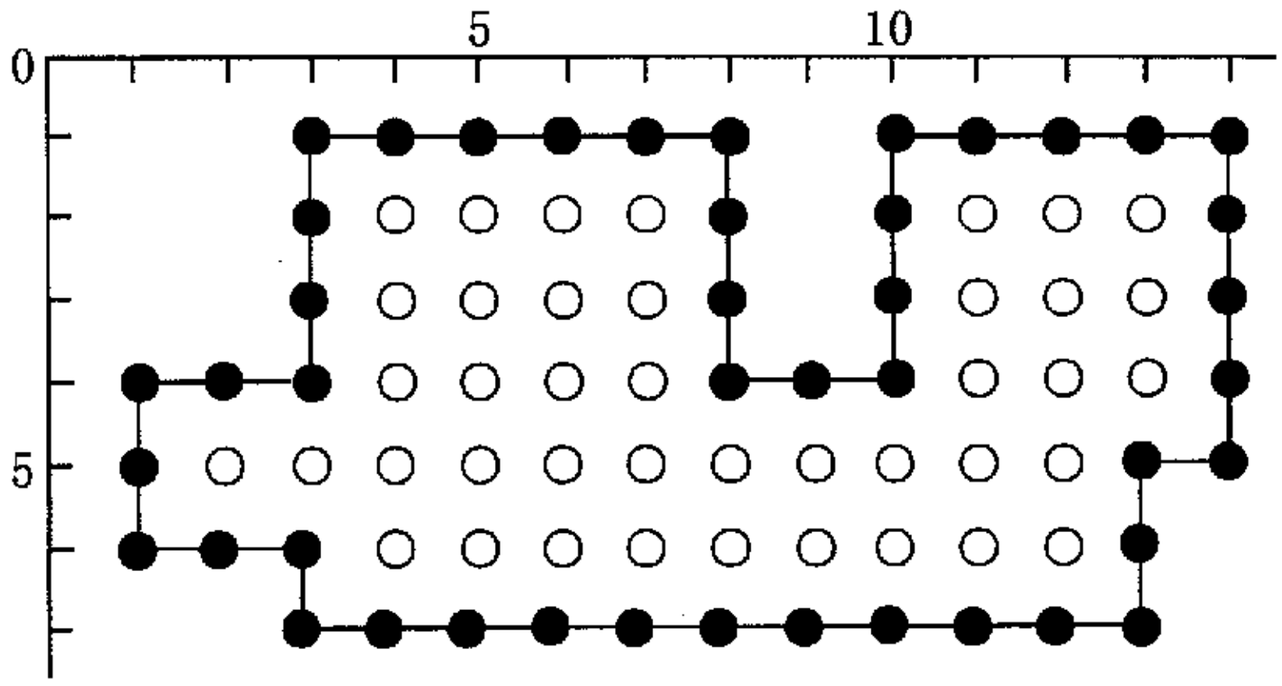


图 2-4 各线段情况

经实验证明采用该算法进行区域填充具有良好的效果。

2.3 采集过程和处理过程的协调

在采集过程中，采集到的图像数据由采集卡控制并通过 PCI 总线送到目的体，由于 PCI 总线传送数据的速度很高，在一帧的时间内除了传送图像数据，还能空出很多时间，用户可以在空出的时间内同时进行图像数据处理。这时，PC 机系统会自动交替分配图像卡与 CPU 的申请，协调完成采集和处理工作。采集过程和处理过程可以顺序进行，也可以并行工作。

1. 采集过程与处理过程顺序进行

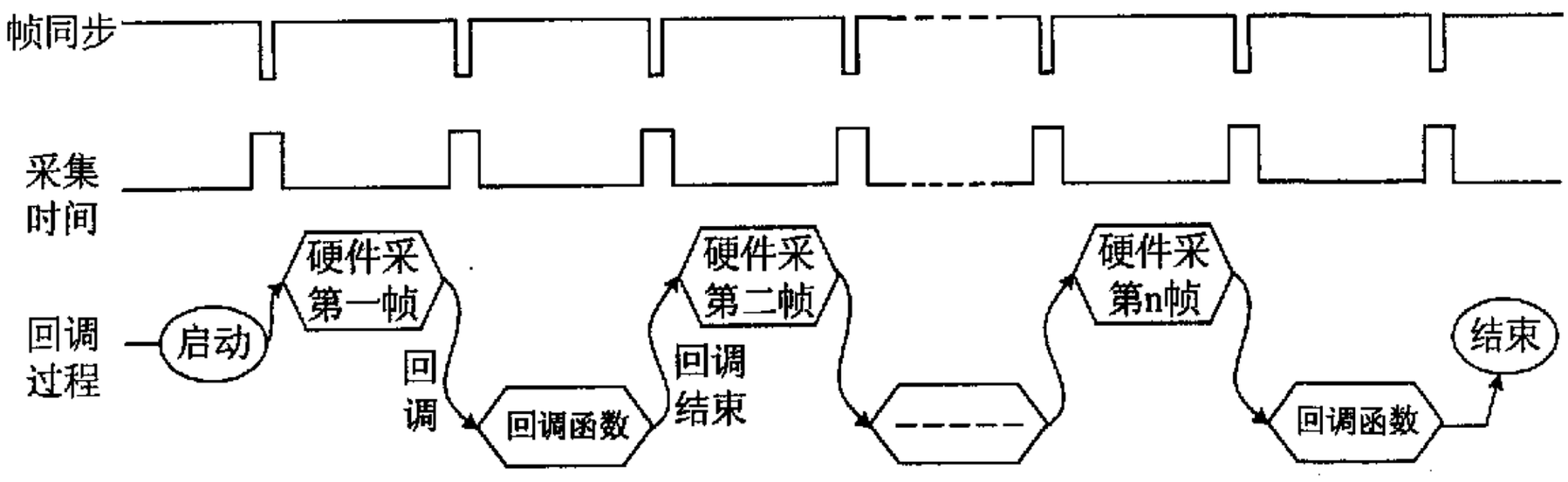


图 2-5 串行工作方式示意图

一般情况下，图像采集和图像处理是按照用户程序顺序进行的，每帧采集结束后，回调用户自己的函数，但不起动下一帧的采集，而是等待回调函数结束，再作下一帧采集，形成采集和用户程序串行的工作方式。该方式适用于采集数据量很大，但对速度和时间没有严格要求的场合。其示意图如图 2-5 所示。

2. 采集过程与处理过程并行工作

在应用中，如果对速度和时间有严格要求，就需要采用采集过程与处理过程并行的工作方式。进行逐帧（或隔帧）的同步并行处理，可以通过 AVICap 窗口类提供的预览回调函数来实现，不等待采集结束就进行回调。由于考虑到一帧图像的采集过程，一般比通过 CPU 读取一帧的图像数据要慢，所以设计的逐帧回调是滞后一帧的。其工作方式示意图如图 2-6 所示。

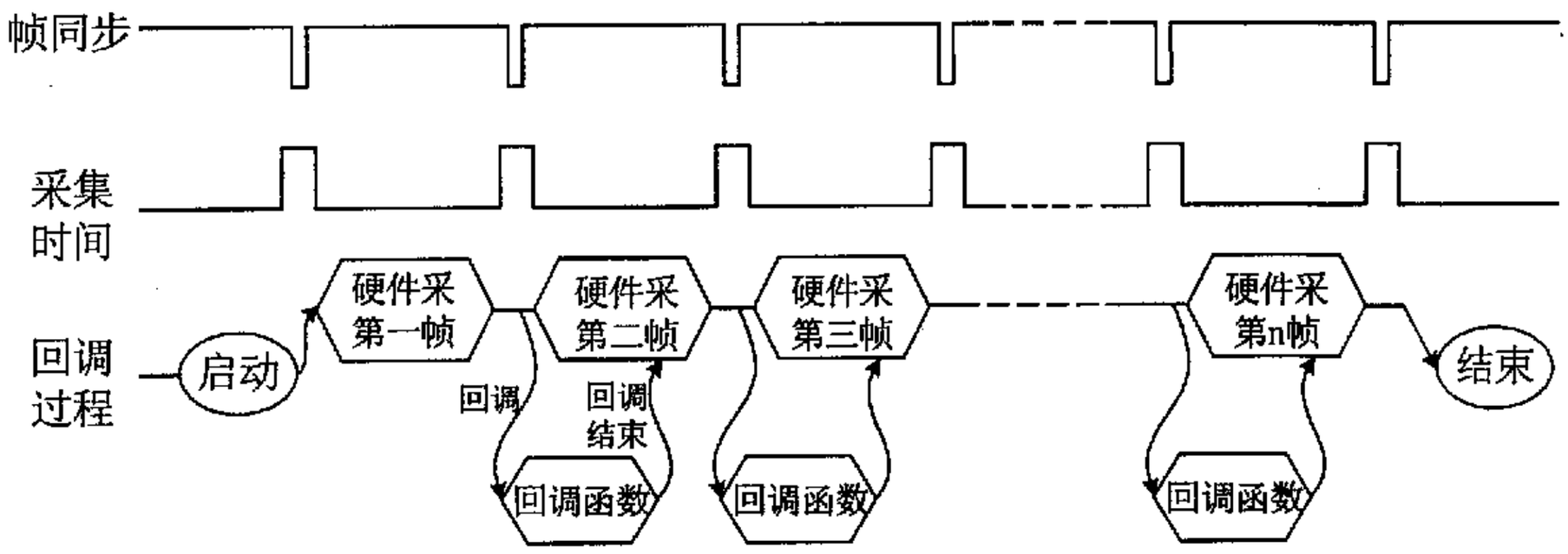


图 2-6 逐帧并行工作方式示意图

本文的程序对速度要求严格，且采集的数据量小、图像数据处理时间短，所以采用逐帧并行的工作方式来实现实时视频采集。

2.4 图像的存储和显示

2.4.1 图像的存储

微机中的图像有 BMP、GIF、JPG 和 TIF 等多种存储方式，其中位图(BMP)格式是微软公司为 Windows 环境设置的标准格式，它是映射到显示像素的位阵列，有 GDI 位图和 DIB 两种类型。GDI 位图又称 DDB，是设备相关的，其位的排列取决于显示设备；而 DIB 是设备无关的，具有更强的灵活性，它既可以存放在内存中，也可以以文件形式保存在磁盘上(BMP 文件)。

位图由如图 2-7 所示的四部分组成，其中 BITMAPFILEHEADER 结构中定义了文件类型、文件长度和位图数据在文件中的起始位置等信息；BITMAPINFOHEA- DER 中定义的是 BMP 图象的宽度、高度、每像素所占 bit 数等内容；RGBQUAD 结构给出了该位图图像使用的调色板，24 位真彩色图和 32 位色图没有颜色表；位图文件的最后一部分，亦即文件的主体，保存的是整幅图象的象素值，它是一个象素阵列。对于内存中 DIB，它包含两部分信息：(1)位图信息(BITMAPINFO)，包括位图信息头和颜色表；(2)位图数据。

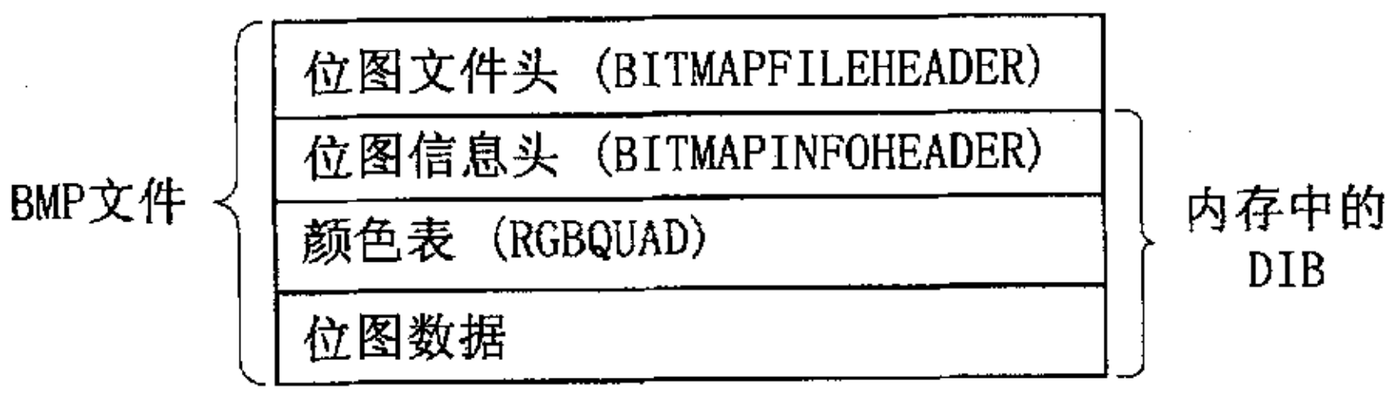


图 2-7 BMP 文件结构示意图

应该注意的是：BMP 存储象素数据是由下而上，由左到右的。并且每行的字节数为 4 的倍数，如果不考虑宽度问题，会得出错误结果。

2.4.2 图像的显示

对图像进行显示，分为两种情况：一是对内存中的图像进行显示，二是对静态图像进行显示。

对内存中的图像显示，直接利用函数 SetDIBitsToDevice( )或 StretchDIBits( )就可以在显示器或打印机上显示，其中 lpbmi 需指向图像数据在内存中的首地址。这两个函数，前者显示时对图像不进行缩放，后者显示时对图像进行缩放。

对静态图像进行显示，就要通过 MFC 类库或采用 WindowsAPI 接口来处理和显示 BMP 位图：首先利用 CFile 类的成员函数打开 BMP 文件并读入内存；然后利用 CPalette 类成员函数 CreatePalette( )创建调色板；最后利用 CDC 类成员函数 SelectObject( )将位图选进兼容 DC、SelectPalette( )选择调色板、RealizePalette( )实现调色板以及 BitBlt( )显示位图。实现过程如下图 2-8 所示。

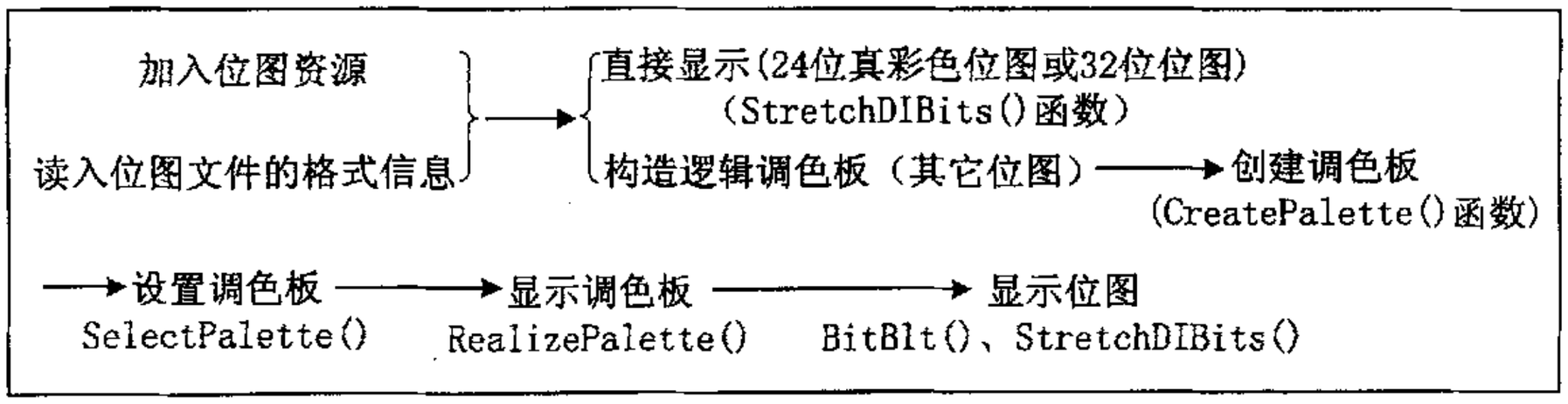


图 2-8 DIB 位图的显示过程

2.5 本章小结

1. 本文采用 VFW 软件包进行图像采集，使程序具有通用性，适合任何视频采集卡；
2. 对摄取图像进行预处理，可以去处图像的噪声，分离物体和背景，增强物体的边缘信息，并且经过预处理的图像，物体边缘点的坐标都被记录下来，为下一章稀疏光流场的计算打下了基础。
3. 采用采集过程与处理过程并行的工作方式，可以满足实时跟踪的要求。

第三章 稀疏光流场的计算

在已知物体表面深度的前提下，恢复物体三维运动参数只需要 5 个光流点，因此本文采用稀疏光流法，仅计算螺栓 6 个特征角点处的光流，克服了经典光流法计算费时，不利于实时跟踪的缺点。进行稀疏光流计算时，首先采用由粗到精两步法提取出特征点，再利用 Hopfield 神经网络实现序列图像间特征点的松弛匹配，最后利用视差计算特征点处的光流。

3.1 经典光流方法

1981 年，Horn 等人在相邻图象间的时间间隔很小，并且图象灰度变化也很小的前提下，推导出灰度图象光流场计算的基本等式。记 t 时刻，场景中物体上某一点 (x, y, t) 在图像平面上的像为 (X, Y)，其灰度值为 $E(X, Y, t)$。在 $t+\Delta t$ 时刻，这一点运动到新位置 $(x+\Delta x, y+\Delta y, t+\Delta t)$，其在图像上的位置变为 $(X+\Delta X, Y+\Delta Y)$，灰度值记为 $E(X+\Delta X, Y+\Delta Y, t+\Delta t)$，假定它与 $E(X, Y, t)$相等，即

$$E(X+\Delta X, Y+\Delta Y, t+\Delta t) = E(X, Y, t) \tag{3-1}$$

将式 (3-1) 左边用泰勒公式展开，经化简和略去两次以上的项，得

$$\frac{\partial E}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}t}+\frac{\partial E}{\partial Y}\frac{\mathrm{d}Y}{\mathrm{d}t}+\frac{\partial E}{\partial t}=0 \tag{3-2}$$

令 $u=\frac{\mathrm{d}X}{\mathrm{d}t}$，$v=\frac{\mathrm{d}Y}{\mathrm{d}t}$，得：

$$E_x u + E_y v + E_t = 0 \tag{3-3}$$

这就是著名的光流约束方程，其中 E_x、E_y、E_t 分别为图像灰度 E 对 X，Y，t 的偏导数。(E_x, E_y) 为图象灰度的空间梯度。由于光流场 (u, v)有两个变量，而基本等式只有一个方程，因此只能求出 (u, v) 沿梯度方向上的值，而不能同时求出 u 和 v。要求解光流 (u, v)，必须引入其他的约束条件。可以使用光滑性约束条件。光流场光滑性的一种度量方式为速度空间梯度幅值的平方。

$$(\frac{\partial u}{\partial X})^2+(\frac{\partial u}{\partial Y})^2 \text{和} \ (\frac{\partial v}{\partial X})^2+(\frac{\partial v}{\partial Y})^2 \tag{3-4}$$

使光流梯度平方和最小用正则化方法，使下式最小

$$\varepsilon^2 = \iint (\varepsilon_1^{\ 2} + \varepsilon_2^{\ 2}) \mathrm{d}X \, \mathrm{d}Y \tag{3-5}$$

式中 $\varepsilon_1 = E_x u + E_y v + E_t$，$\varepsilon_2 = (\frac{\partial u}{\partial X})^2 + (\frac{\partial u}{\partial Y})^2 + (\frac{\partial v}{\partial X})^2 + (\frac{\partial v}{\partial Y})^2$ (3-6)

采用松弛算法求解，得到递推公式：

$$u(m+1) = \bar{u}(m) - E_x (E_x \bar{u}(m) + E_y \bar{v}(m) + E_t) / (\lambda^2 + E_x^2 + E_y^2) \tag{3-7}$$

$$v(m+1) = \bar{v}(m) - E_y (E_x \bar{u}(m) + E_y \bar{v}(m) + E_t) / (\lambda^2 + E_x^2 + E_y^2) \tag{3-8}$$

式中 m 表示第 m 次迭代，而 $\bar{u}(m)$，$\bar{v}(m)$ 为第 m 次迭代后像素点速度 $u(m)$，$v(m)$的邻域平均值。

以上介绍的光流算法，计算的是图像中每个像素点的光流，因此又称为稠密光流法。虽然基于光流的方法不需要进行连续图像间特征的匹配，根据假设和一些其它的信息就可以计算出物体的运动和结构，但光流计算在理论上和实际上仍存在许多不足之处。

(1) 只适用于短时间小运动量的处理　光流计算的基本等式是假定相邻两幅图像中对应两点的灰度值不变而得到的。在这个假设条件下，首先要求相邻两幅图像间隔时间 t 比较短，要求速度必须小于 20 个像素/帧；

(2) 计算速度慢，误差较大[29]　计算时忽略了两次以上的项，只用到了线性项，只有当两次以上的项相对于一次项任意小时，计算结果才可靠，否则有较大误差。更重要的是该算法是在整幅图像平面上进行的，并且用迭代公式，算法费时，不适合于实时跟踪。例如：Horn and Schunck 计算 100%的稠密光流大约需要 8 分钟的时间，错误率为 33.4%；Fleet and Jepson 计算 76%的稠密光流大约需要 30 分钟的时间，错误率为 0.36%。

(3) 易受噪声干扰　该算法用到了微分运算，而微分运算是对噪声敏感的，阶数越高，对噪声越敏感。

3.2 稀疏光流场方法

由于经典光流法具有计算速度慢等缺点，无法满足实时跟踪的要求，并且考虑到估计物体运动所需特征点数量的要求，我们没有必要计算出图像中各点的光流值，只要计算物体上稀疏 (sparse) 特征点处的光流即可。本文提出了一种稀疏光流法，它的基本思想是：首先利用基于特征的方法在一幅图像上抽取相对稀疏、但有明显差别的角点，并在图 像之间建立起这些特征点的对应；其

次利用视差计算出特征点的光流，并利用稀疏光流进行物体运动参数和结构参数的估计。

采用稀疏光流法具有以下优点：

（4）对噪声不敏感　计算没有用到微分运算，只要特征点的提取较准确，就不会产生噪声；

（5）运算速度快　经典光流法是在整幅图像平面上进行的，并且用到了迭代算法，而稀疏光流法仅计算特征点的光流，并且特征点的匹配只需进行一次，所以计算速度快。

（6）可以计算帧间大的视差运动，边缘不被模糊，并且比较适合多目标的场合。

稀疏光流场计算的关键是进行特征点的提取和图像间特征点的匹配。

3.2.1 特征点的提取

轮廓特征点包括角点、切点和拐点，它们决定了图象中目标的形状，所以在图象匹配、目标描述与识别以及运动估计、目标跟踪等领域，轮廓特征点的提取具有十分重要的意义。

本文要提取的是近似正六边形的角点。在计算机视觉和图象处理中，对于角点的定义有不同的表述，如：图象边界上曲率变化明显的点；图象边界方向变化不连续的点；图象中梯度值和梯度变化率都很高的点等等。角点检测的方法也不尽相同，如用链码跟踪后的轮廓点计算曲率来判定角点；利用方向导数来检测角点；直接以图象灰度信息来检测角点等等。对于已有的主要的角点检测方法，可以把它们归为两类：一是根据图象边缘特征，用轮廓点来计算边缘曲率或夹角来判定角点；二是直接利用灰度信息进行角点检测。

但是直接利用灰度信息进行角点检测对噪声敏感，精度较低；若直接利用轮廓点计算曲率或夹角来判定角点，算法费时，精度不高。在稀疏光流计算中，特征点的提取是最耗时的，要满足实时跟踪的要求就必须加快特征点的提取速度。 因此本文提出一种综合的算法：首先利用灰度信息提取出候选角点。经过预处理的图像对噪声不敏感，并且只有 0 和 255 两种灰度级，容易进行角点的粗提取，通过第一步的提取，使要处理的角点数远远小于轮廓点的数目，加快了计算速度；然后利用候选角点组成的直线的夹角对这些候选角点进行取舍，筛选出真正的角点。

1. 候选角点的提取

设有一个以轮廓点为中心、半径为 R 的圆盘在轮廓线上移动，如图 3-1 所示。当圆盘位于 A 位置时，目标和背景在圆盘中的面积各是圆盘面积的一半；当圆盘位于 B、C 位置时，目标在圆盘中的面积小于一半，背景在圆盘中的面积大于一半；当圆盘位于 D 位置时，目标在圆盘中的面积大于一半而背景在圆盘中的面积小于一半。图 3-1 中 B、C、D 的位置是角点的位置。基于上述观察，可有如下结论：当圆盘处在直线上时，目标和背景在圆盘中的面积各是圆盘面积的一半；当角点位于圆盘中时，目标和背景在圆盘中的面积总有一个小于圆盘面积的一半，根据这一点可以进行角点检测。

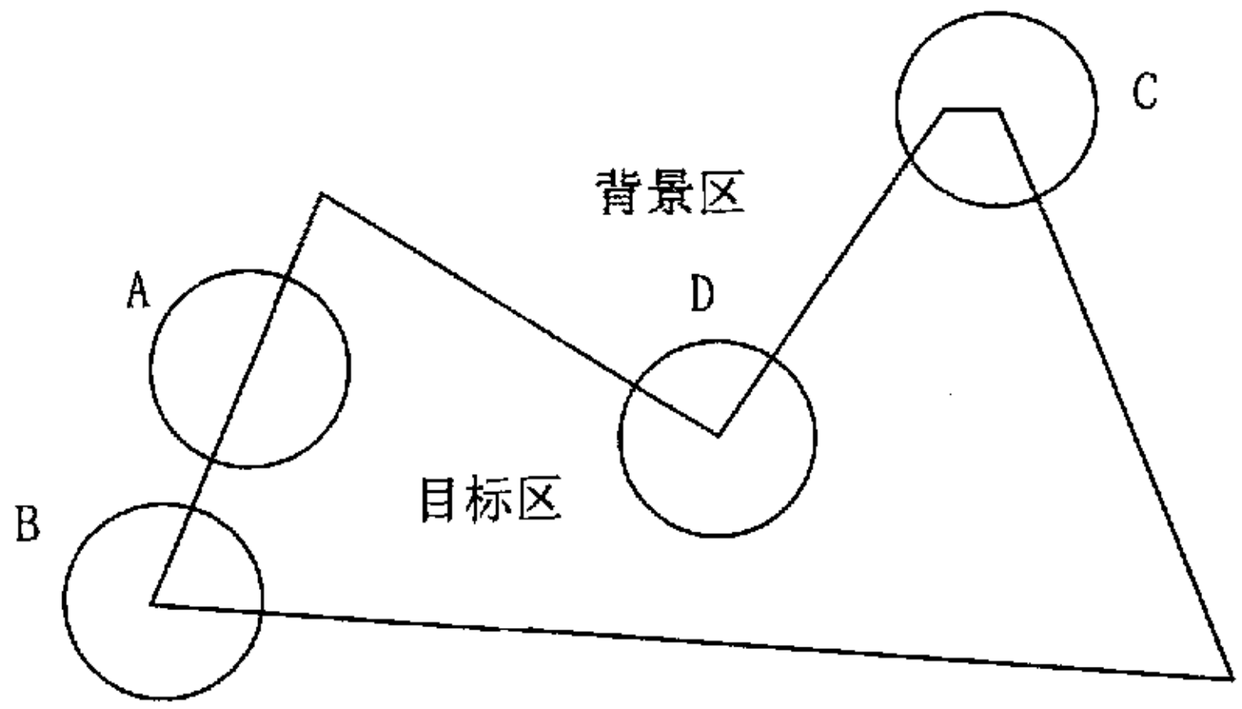


图 3-1 圆盘在轮廓线上移动示意图

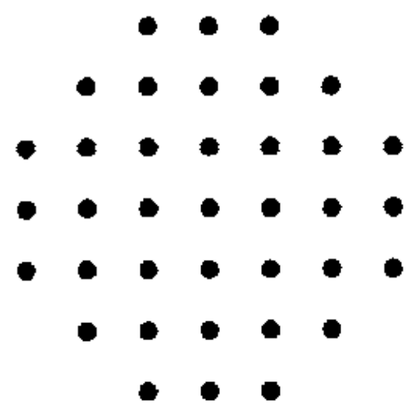


图 3-2　37 点阵的近似圆

经过种子填充后的图像，边缘以及目标区的灰度值都为 0（黑色），而目标区之外的区域的灰度值为 255（白色）。本文采用 37 点阵的近似圆（分布为 3577753）如上图 3-2 所示，让该圆以预处理图像轮廓边缘上的点为圆心，并在轮廓边缘点上依次滚动。若该圆所包围的区域中灰度值为 0 的点的个数位于 18 和 23 之间时，我们就把该点作为候选角点。具体编程实现为：

……

```
for(j=a[k]-3; j<=a[k]+3; j++)
    {for(i=b[k]-connum; i<=b[k]+connum; i++)
     { lpsrc = (unsigned char*)lpvBits + linebyte * j + i*4;
      //指向源图像倒数第 j 行，第 i 个象素的指针(对 32 位的图像进行操作)
        pixel = (unsigned char)*lpsrc;    // 取得当前指针处的像素值
        if(pixel == 0)
            nCount++;     // 如果为黑色的点，nCount 加 1
       }
     ......
    }
if(nCount<23&&nCount>18)        // 当黑色的点的个数位于 18 和 23 之间时
    { lpsrc1=(unsigned char*)lpvBits + linebyte * a[k]+b[k]*4;
     *lpsrc1=10;                          // 把候选角点的灰度值置为 10
      *(lpsrc1+1)=10;
     *(lpsrc1+2)=10;
      ta1[w]=a[k];                        // 候选角点的坐标值
      tb1[w]=b[k];
      w++;
      ......
    }
```

2. 伪角点的去除

得到候选角点后，还需要剔除其中的伪角点。我们根据三个角点组成的两直线夹角余弦值的大小来进行取舍。如图 3-3 所示，设相邻三点 A、B、C 的坐标值分别为 (a_1, b_1)，(a_2, b_2)和(a_3, b_3)，则直线 AB 和 BC 所成夹角的余弦值为：

$$\cos\theta = \frac{(a_1-a_2)^2+(b_1-b_2)^2+(a_3-a_2)^2+(b_3-b_2)^2-(a_1-a_3)^2-(b_1-b_3)^2}{2\sqrt{(a_1-a_2)^2+(b_1-b_2)^2}\cdot\sqrt{(a_3-a_2)^2+(b_3-b_2)^2}} \tag{3-9}$$

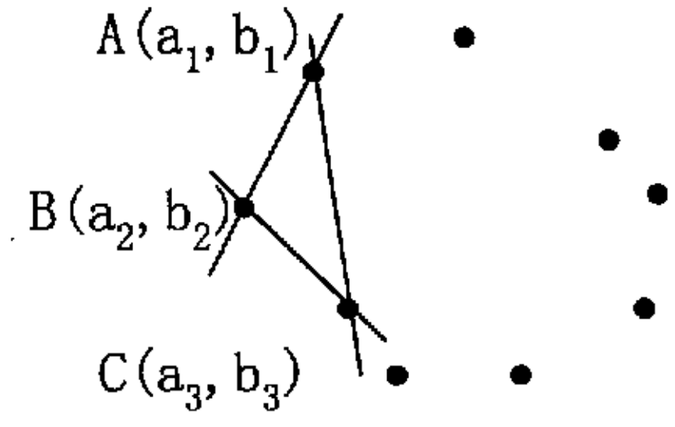


图 3-3 利用夹角判定角点示意图

若 $|\cos\theta| > \cos(\pi/\mathrm{div})$，则 B 点应剔除。其中 π /div 为阈值，首先给 div 赋一个初值，随着循环次数的增加，div 的值有规律的减小。流程图如图 3-4 所示：

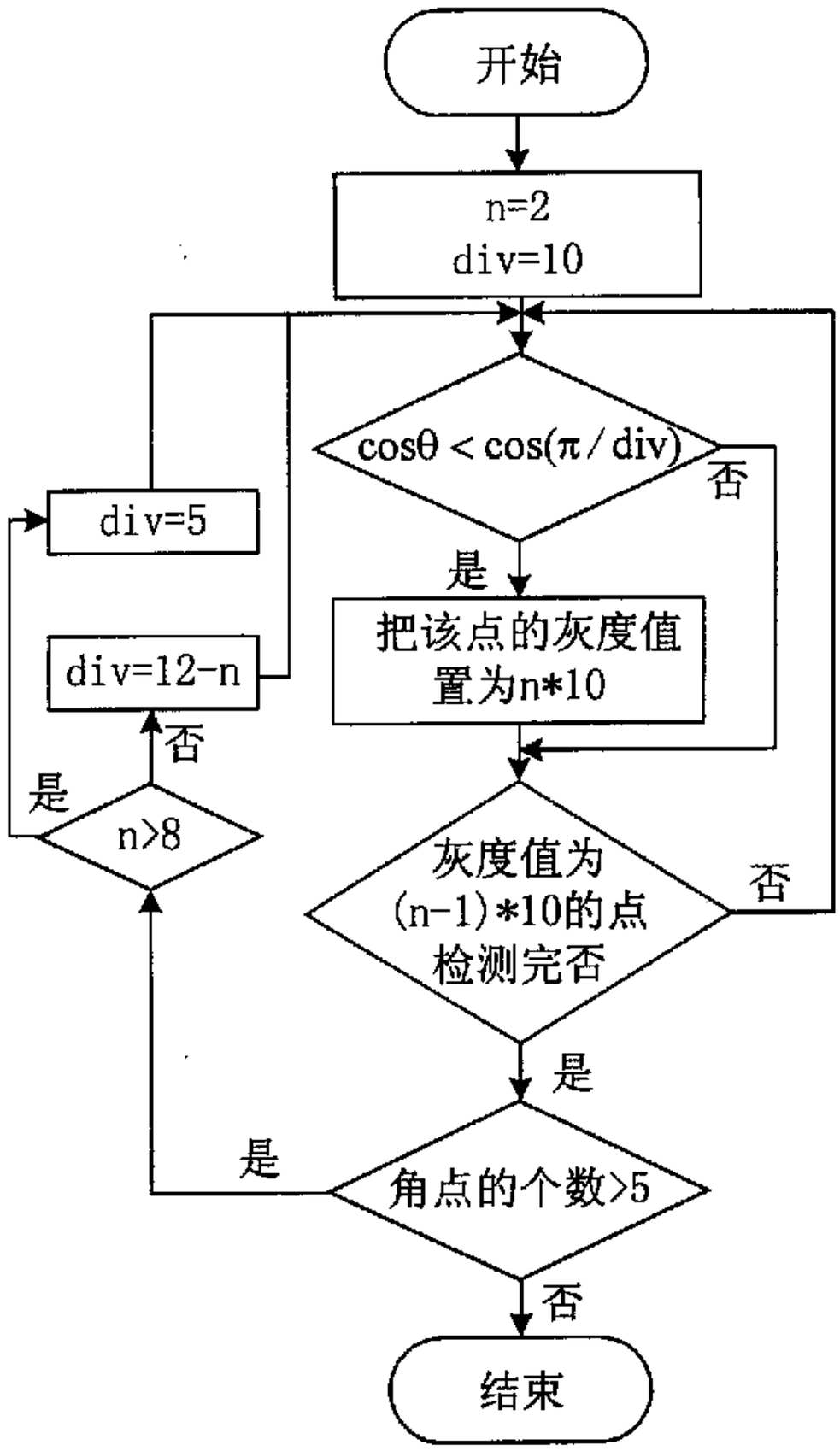


图 3-4 舍弃伪角点流程图

3.2.2 图像间特征点的匹配

特征点匹配问题可以这样描述：已知两个点集 $P=\{p_1, p_2, \cdots, p_m\}$及 $Q=\{q_1, q_2, \cdots, q_n\}$，匹配要实现的目的就是确定两个点集之间的对应关系。它在配准、导航、变化检测及立体映射等应用中是一项极其重要的技术，已引起大量研究人员的普遍关注，目前已发表了许多文献，提出了各种匹配方法，大致可以分为：基于聚类的特征点匹配[30]、基于图论的特征点匹配[31]和基于松弛迭代的特征点匹配[32]。

特征点的匹配速度和匹配精度在很大程度上制约着稀疏光流的计算速度和精度，以上介绍的匹配方法不是侧重匹配速度，就是侧重匹配精度，因此本文采用 Hopfield 神经网络完成特征点的松弛匹配过程，该方法把具有强大的并行处理能力和稳定性的神经网络和匹配精度较高的松弛匹配方法相结合，加快了匹配速度，提高了匹配的稳定性和精度。

1. 基于松弛迭代的特征点匹配

设有两特征点集 $P=\{p_1, p_2, \cdots, p_m\}$及 $Q=\{q_1, q_2, \cdots, q_n\}$。对每一个点对$(p_i, q_j)$，定义两个特征点集间的相对偏移，令 $\delta ij(h, k)$为当 p_i与 q_j相匹配时 p_h与 q_k间的距离，即：

$$\delta_{ij}(h,k)=\left\|(p_h-p_i)-(q_k-q_j)\right\| \tag{3-10}$$

假如$\left|\delta_{ij}(h,k)\right|$为零，则表示 q_k相对于 q_j等同于 p_h相对于 p_i，因此点对(p_h, q_k) 应当给予(p_i, q_j) 以最大的支持。随着$\left|\delta_{ij}(h,k)\right|$的增加，其支持度应减小。于是，令$(p_h, q_k)$ 对(p_i, q_j) 的支持度为

$$\phi(\left|\delta_{ij}(h,k)\right|)=\frac{1}{1+\left|\delta_{ij}(h,k)\right|^2} \tag{3-11}$$

若要求当 p_i与 q_j配对时，p_h仅与一个 q_k相配对，即与 p_h相联系的、对(p_i, q_j) 的支持度最大的 q_k相配对，则可得到如下的支持度表达式：

$$\max_{k\neq j}\phi(\left|\delta_{ij}(h,k)\right|) \tag{3-12}$$

为了得到(p_i, q_j)的初始支持度，我们取所有 p_h之和的平均：

$$S^0(p_i,q_j)=\frac{1}{m-1}\sum_{h\neq i}\max_{k\neq j}\phi(\left|\delta_{ij}(h,k)\right|) \tag{3-13}$$

在计算 $S^0(p_i, q_j)$时，我们平等地看待每一个点对(p_h, q_k)，因为在无其它先

验知识的条件下，任意两个点均可以配对。但是，在第 r 次迭代（$r>0$）时，(p_h, q_k)对(p_i, q_j)的支持度不仅依赖于 p_h 与 q_k 间的位置差别，而且也依赖于它们的 $S^{r-1}(p_h, q_k)$值，即允许局部支持度的反馈。这两个因素可以不同的方式结合在一起，这里我们取它们中的最小值，因此有：

$$S^r(p_i,q_j)=\frac{1}{m-1}\times\sum_{h\neq i}\max_{k\neq j}\min[S^{r-1}(p_h,q_k),\phi(\left|\delta_{ij}(h,k)\right|)] \tag{3-14}$$

该迭代一直进行到对每一 p_i 除最可能的点对以外其余所有点对的支持度均小于给定的门限。

文献[32]已经表明该方法具有一定的抗旋转及比例变化的能力。

2. Hopfield 神经网络

Hopfield 神经网络是由 Hopfield 于 1982 年提出的一种网络模型，也是迄今得到最广泛应用的神经网络模型之一。它的成功之处在于，定义了网络稳定性判据——能量函数（也称李雅普诺夫函数）的概念，这种网络在联想存储及优化计算等领域得到了成功的应用。该网络是一个单层相互连接型神经网络模型，每一个神经元都与其它的神经元通过反馈的方式互相连接，其网络结构示意图如图 3-5 所示。

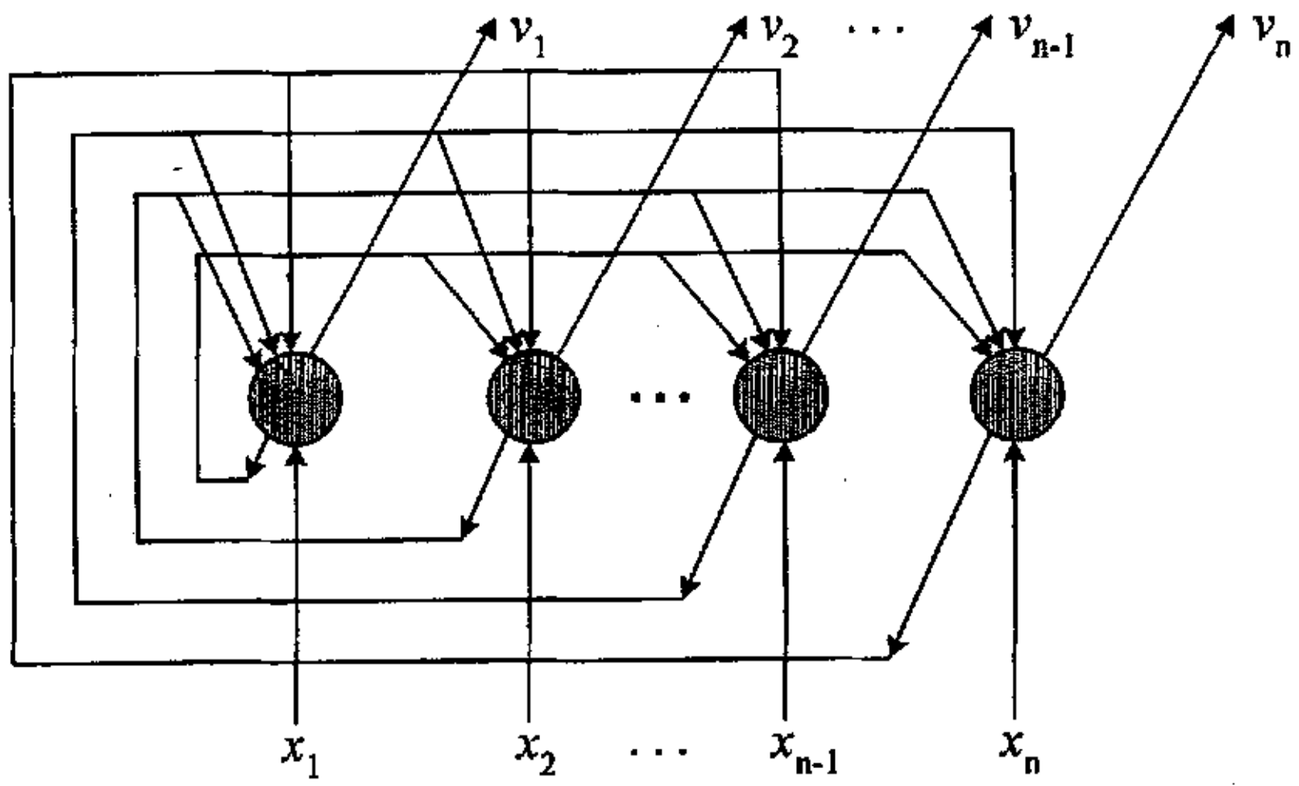


图 3-5 Hopfield 网络结构示意图

Hopfield 网络是一个动力学系统，一旦网络的连接权矩阵学习形成，只要输入某个模式样本，网络就不断演化，直至系统达到状态空间的定态。该网络具有很好的稳定性和强大的并行处理能力。

若令 u_i 表示神经元 i 的状态，则描述 u_i 变化的方程为

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = -\frac{u_i}{\tau_i} + \sum_{j \neq i} T_{ij} V_j + I_i \tag{3-15}$$

式中 T_{ij} —— 为神经元 i 与 j 间的连接权；

I_i —— 表示外部输入偏置；

V_j —— 表示神经元 j 的输出；

τ_i —— 为时间常数，为简单起见，把 τ_i 设定为 1；

V_j 的取值为：

$$V_i = g(u_i) = \frac{1}{2}(1 + \tanh(\lambda u_i)) \tag{3-16}$$

它是一个 S 型单调递增函数。

研究表明，当连接权矩阵无自连接以及具有对称性，即

$$T_{ii} = 0 \quad (i = 1, 2, \cdots, \mathrm{N}) \tag{3-17}$$

$$T_{ij} = T_{ji} \quad (i, j = 1, 2, \cdots, \mathrm{N}) \tag{3-18}$$

时，算法是收敛的。

描述网络状态的能量函数为

$$E = -\frac{1}{2}\sum_i \sum_{j \neq i} T_{ij} V_i V_j - \sum_i V_i I_i + \sum_i \frac{1}{\tau} \int_0^{V_i} \mathrm{g}^{-1}(t)\, dt \tag{3-19}$$

由于神经元 i 的输出变化 dV_i 而带来的 E 的变化 dE_i 为

$$dE_i = -[\sum_{j \neq i} T_{ij} V_j + I_i - \frac{u_i}{\tau_i}] dV_i = -\frac{du_i}{dt} \tag{3-20}$$

3. 用 Hopfield 神经网络实现松弛匹配的算法

在松弛匹配算法中，p_i 与 q_j 配对是在将 P 中的其余点映射到 Q 时的吻合度之和达到最大值的约束条件下进行的，在该过程中需要利用 $S(p_h, q_k)$ 及 $\phi(|\delta_{ij}(h,k)|)$ 调整 $S(p_i, q_j)$，这个过程是并行的。若能找到一个评价当前状态与约束间一致性的优度函数，那么松弛过程就可看作是通过迭代以增加优度函数值的过程。

Hopfield 可以看作是一种并行的计算模型；并且在网络演变过程中，倾向于使能量函数达到最小的稳定状态。从这个观点上来看，Hopfield 神经网络非常适用于实现松弛过程。只要能找到一个优度函数，使得其值在松弛过程中不断增加，并将该优度函数变换成 Hopfield 神经网络的能量函数，网络中的一个神经

元与一个 $S(p_i, q_j)$值相对应，而神经元之间的连接则体现了各神经元之间的关系。

在我们的松弛匹配过程中，定义优度函数 F 如下：

$$F=\sum_i\sum_j S(p_i,q_j)\frac{1}{m-1}\sum_{h\neq i}[\max_{k\neq j}S(p_h,q_k)+\max_{k\neq j}\phi(\left|\delta_{ij}(h,k)\right|)] \tag{3-21}$$

并使用如下的 Hopfield 神经网络能量函数：

$$E=-\frac{m-1}{2}\times F+\frac{1}{2}\sum_i[\sum_j S(q_i,q_j)-1]^2 \tag{3-22}$$

其中第 1 项除了存在一个常数值$(m\text{-}1)/2$ 外与式(3-21)基本相同；而第 2 项在所有表示某个 p_i 不同配对点的神经元的输出之和等于 1（这意味着每一个 p_i 只能与一个 q_j 配对）时，得到最小值零。

将式(3-21)代入式(3-22)，并展开式，得：

$$E=-\frac{1}{2}\sum_i\sum_j\sum_{h\neq i}\max_{k\neq j}S(p_i,q_j)S(p_h,q_k)-\frac{1}{2}\sum_i\sum_j\sum_{h\neq i}\max_{k\neq j}S(p_i,q_j)\phi(\left|\delta_{ij}(h,k)\right|)$$

$$+\frac{1}{2}\sum_i(\sum_j S(p_i,q_j))^2-\sum_i\sum_j S(p_i,q_j)+\frac{m}{2} \tag{3-23}$$

忽略常数项，由式(3-23)可得连接权矩阵和输入偏置的值为

$$T_{(ij)(hk)}=T'_{(ij)(hk)}-\sigma_{(ij)(hk)} \tag{3-24}$$

其中

$$T'_{(ij)(hk)}=\begin{cases}1 & (i\neq h,j\neq k,S(p_h,q_k)=\max_{r\neq j}S(p_h,q_r))\\ 0 & (\text{其它})\end{cases} \tag{3-25}$$

$$\sigma_{(ij)(hk)}=\begin{cases}1 & (i=h)\\ 0 & (i\neq h)\end{cases} \tag{3-26}$$

而
$$I_{ij}=\frac{1}{2}\sum_{h\neq i}\max_{k\neq j}\phi(\left|\delta_{ij}(h,k)\right|)+1 \tag{3-27}$$

注意这里（ij）和（hk）表示两个单一的下标。

为了保证映射后 Hopfield 网络的收敛性，T 必需为对称矩阵，且对角线上元素为零，因此我们将 $T_{(ij)(hk)}$和 $\sigma_{(ij)(hk)}$的定义修改为：

$$T^*_{(ij)(hk)}=\frac{1}{2}(T'_{(ij)(hk)}+T'_{(hk)(ij)}) \tag{3-28}$$

$$\sigma^*_{(ij)(hk)}=\begin{cases}1 & (i=h,j\neq k)\\ 0 & (\text{其它})\end{cases} \tag{3-29}$$

此时连接权矩阵为

$$T_{(ij)(hk)} = T^*_{(ij)(hk)} - \sigma^*_{(ij)(hk)} \tag{3-30}$$

当 $T_{(ij)(hk)}$和 I_{ij} 固定以后，神经网络的最终状态便仅仅依赖于它的初始状态。因此在网络演变开始以前，必须根据先验知识设定各神经元的初始状态。在本文中，设定各神经元的初始状态u_{ij}^0为

$$u_{ij}^0 = g^{-1}(S^0(p_i, q_j)) \tag{3-31}$$

在确定了连接权矩阵、输入偏置及各神经元的初始状态以后，网络的能量面和局部最小点便完全确定了，整个网络将达到某个特定的局部最小点。

本文的图像采集速度为 1 秒钟 12 帧，并且物体运动速度不大，因此该算法能够满足要求。

3.2.3 稀疏光流场的计算

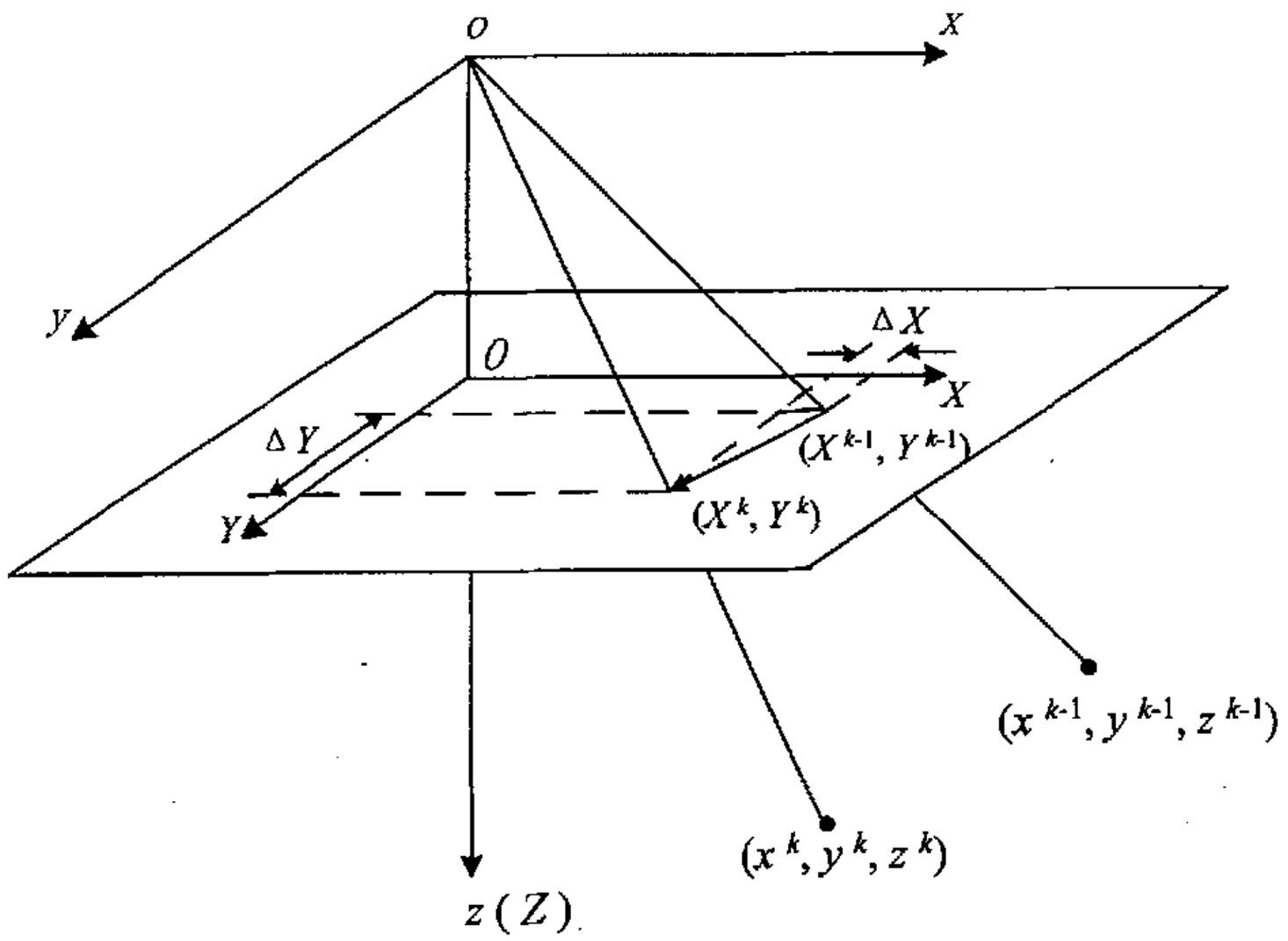


图 3-6 物体的三维运动和图像平面

如图 3-6，空间物体点 P 从（x^{k-1}，y^{k-1}，z^{k-1}）运动到（x^k，y^k，z^k），时间间隔为$\Delta t_k(\Delta t_k = t_k - t_{k-1})$，反映在图像平面上，该点从（$X^{k-1}$，$Y^{k-1}$）移动到（$X^k$，$Y^k$），其位移量为（ ΔX^k，ΔY^k），其中

$$\begin{cases} \Delta X^k = X^k - X^{k-1} \\ \Delta Y^k = Y^k - Y^{k-1} \end{cases} \tag{3-32}$$

我们称物体表面上的点在相邻两时刻图像上相应的位置差为视差，视差的估计也称为对应问题。视差除以时间间隔Δt_k称为瞬时位置速度。在图像序列反映的景物范围中，各物体或分部的运动是不同的，形成众多的瞬时速度向量，这些不同的瞬时位置速度向量分布在图像上形成的向量场，称为光流场。

利用前几节的方法提取出特征点，并在相继两幅图像间进行特征点匹配之后，就可利用视差计算出稀疏光流场。每个特征点处光流计算公式为：

$$\begin{cases} u = \dfrac{\Delta X^k}{\Delta t_k} \\ v = \dfrac{\Delta Y^k}{\Delta t_k} \end{cases} \tag{3-33}$$

3.3 本章小节

1. 考虑到实时跟踪对计算速度的要求和估计物体运动所需特征点数量的要求，本文采用稀疏光流法，仅计算螺栓端面边界六个顶点处的光流；
2. 采用面积约束和夹角约束相配合的方法进行特征点的提取，运算量小，计算速度快。
3. 本文采用 Hopfield 神经网络实现序列图像间特征点的松弛匹配，把松弛匹配的准确性和 Hopfield 神经网络强大的并行处理能力结合起来，快速而有效地实现了特征点匹配。

第四章 物体三维运动参数和结构参数的估计

利用运动物体在二维图像平面上的投影坐标及平面光流来恢复物体的三维结构参数和运动参数，是序列图像分析的最终目的。物体结构参数中最重要的参数之一是深度信息，即物体和摄像机之间的距离，但单目摄像机获取深度较为困难。本文采用散焦法仅需一幅图像就可获取深度信息，从而克服了用单目视觉获取深度信息的困难，并且该方法计算量小、测试效率高，能够满足实时跟踪的要求。在已知表面深度的情况下，利用 5 个光流点即可求得物体的三维运动参数。

4.1 物体深度信息估计

4.1.1 聚焦法和散焦法

我们采用的摄像机系统是由光学镜头组、CCD 感应阵列和 A/D 变换器等子系统组成。摄像机完成从三维光学场到两维电场的变换。镜头组可以等效为一个薄透镜。图 4-1 为单目视觉方法成像的几何图示：

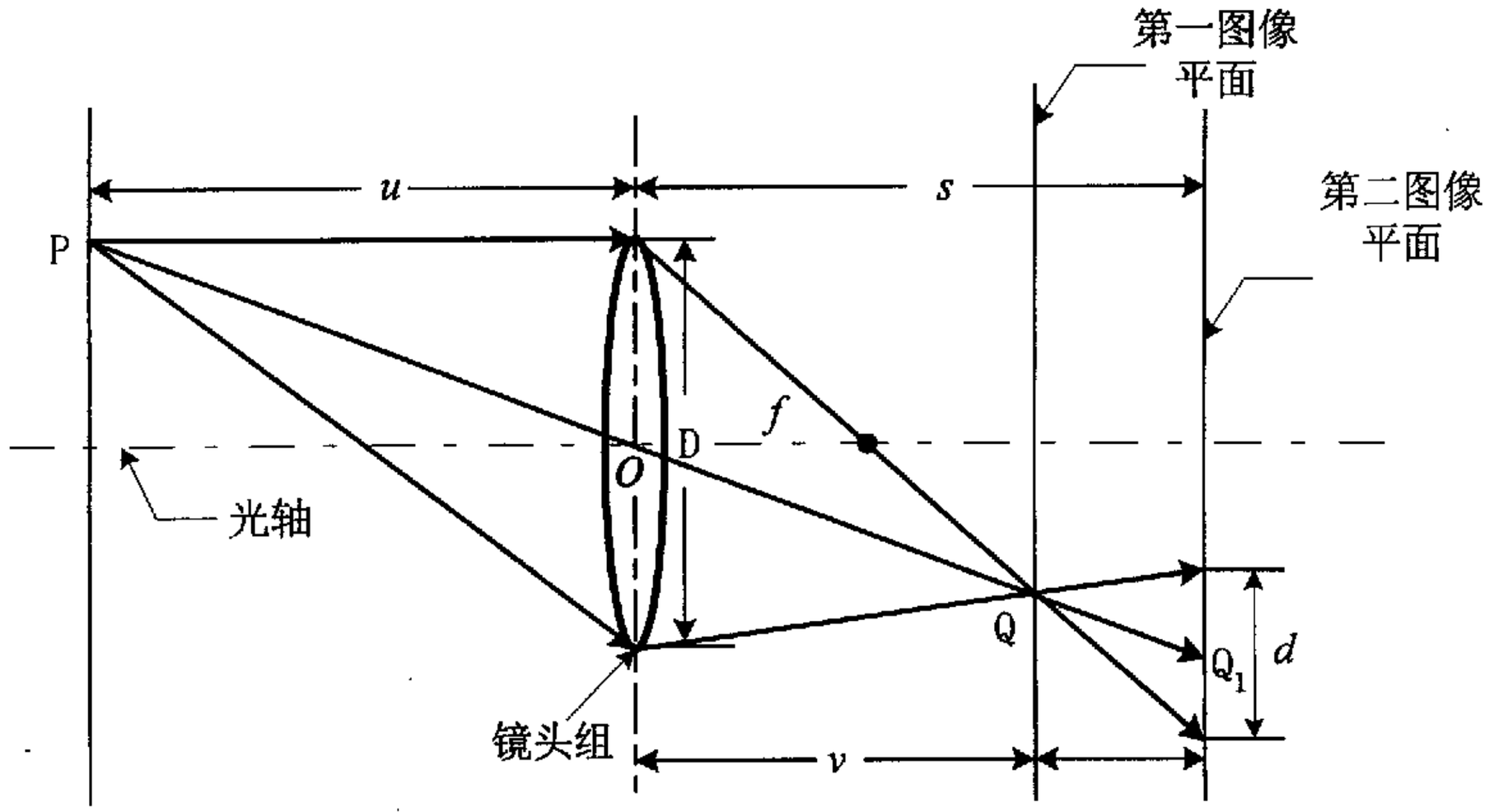


图 4-1 成像几何图示

令 P 为物体表面上的一点，Q 表示 P 点所成的聚焦像点，Q_1 表示 P 点成散焦图像时弥散圆的圆心，f 表示透镜的焦距，u 表示物距，v 表示 P 点成聚焦图

像时的像距，s 表示 P 点成散焦图像时的像距。

在图 4-1 中，P 点和 Q 点应该满足透镜成像公式：

$$\frac{1}{f}=\frac{1}{u}+\frac{1}{v} \tag{4-1}$$

若物距 u 为已知，把图像平面置于第一图像平面处，可以得到清晰的图像。这个图像称为聚焦图像。反过来，通过调整图像平面的位置和焦距，当在图像平面上得到清晰的图像时，可以容易地由式 (4-1) 计算出物距来。这种由聚焦图像求取物点深度的方法称为由聚焦求取深度(depth from focus)。

在图 4-1 中，若物点位置不变，而图像平面在聚焦位置两侧移动，则物点 P 在图像平面上的投影不再是一个点，而是一个圆斑。当图像平面位于第二图像平面时，圆斑直径为 d，根据相似三角形关系和式 (4-1)，可求得：

$$d=s\cdot D\left(\frac{1}{f}-\frac{1}{u}-\frac{1}{s}\right) \tag{4-2}$$

式中 D 为透镜组孔径直径。

另外，假设 σ 是对应于圆斑的点扩展函数二维高斯分布的扩展参数，它与圆斑直径的关系可以表示为 $d=k\sigma$ (其中 k 是一个需要标定的常数，且其随着光学系统的不同而变化)。而透镜组孔径直径 D 可以表示为 $D=f/F$，其中 F 为透镜的光圈指数。此时可以求得物距 u 为：

$$u=\frac{sf}{s-f-Fk\sigma} \tag{4-3}$$

由式 (4-3) 可知，如果已知摄像机的光学系统参数(如 F 、s、f)和 σ 以及 k，则可以求得物距 u。其中 s 和 f 可由摄像机标定过程决定。这种由单目摄像机获取的一幅散焦图像来求取深度的方法称为由散焦求取深度(depth from defocus)。

理论上，当图像平面处于位置 v 时获得聚焦图像，当图像平面处于其他位置时都为散焦图像。应当定义一个判据来区分聚焦图像和散焦图像。由于散焦图像丢失了部分高频能量，用傅立叶变换和频谱分析方法来得到这样的判据，似乎应该是首选的。然而，频谱分析计算量比较大，以目前计算机的运算速度还不能满足实时控制的要求。目前使用较多的为空间域判据。

对应于高通滤波器的空间域方法为微分算子，如拉普拉斯算子。

$$\nabla^2 I=\frac{\partial^2 I}{\partial x^2}+\frac{\partial^2 I}{\partial y^2} \tag{4-4}$$

对整幅图像进行拉普拉斯变换后，对其绝对值求和，而聚焦图像具有最大值。

另一种常用的判据基于如下事实，清晰的图像反映在灰度直方图为剧烈的变化，而模糊的图像其灰度直方图的变化是渐缓的。散焦减小了灰度的波动量。因而

$$\rho^2 = \frac{1}{N^2}\sum_N\sum_N (I(x,y)-\mu)^2 \tag{4-5}$$

式中 μ 为灰度分布的平均值，最大的 ρ^2 值对应于清晰的聚焦图像。这种度量对应于灰度分布功率谱的积分。

用聚焦法来求深度，虽然精度高，但需要的图像数量多，计算时间长，效率低，难以满足实时跟踪的要求；而用散焦法则只需要一幅图像就可以求深度，因此计算量小、测量效率高，缺点是测量精度较聚焦法要低，本文采用 4.1.3 节的深度修正法，可以获得较高的精度。

4.1.2 散焦边缘

实际应用中，孤立物点是不存在的。由于物体表面点的连续性，物体所对应图像其域内点的散焦性也难以分析，而且在图像中边缘的模糊性是很明显的，因此本文采用了一种局部直线深度求取的通用方法。如图 4-2 所示：

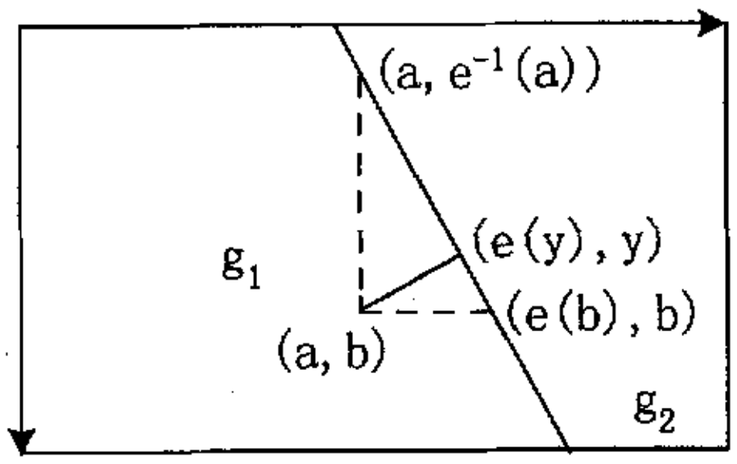


图 4-2 任意直线的边缘和邻点

设有一阶跃型边缘，该边缘是一局部直线边缘，直线方程为：

$$x = e(y) = h_1 y + h_2 \tag{4-6}$$

直线两侧的区域分别为 Ω_1 和 Ω_2，Ω_1、Ω_2 域内的灰度分别是 g_1 和 g_2。这条局部直线边缘由于散焦而发生模糊。这个散焦图像为清晰的聚焦图像与点扩展函数的卷积。在边缘临近一点(a，b)处的灰度值为：

$$g(a,b)=g_1\int_{-\infty}^{+\infty}\int_{-\infty}^{e(y)}G(\sqrt{(x-a)^2+(y-b)^2},\sigma)\,dxdy + g_2\int_{-\infty}^{+\infty}\int_{e(y)}^{+\infty}G(\sqrt{(x-a)^2+(y-b)^2},\sigma)\,dxdy \tag{4-7}$$

把直线方程代入，经整理后得到

$$g(a,b)=g_1 N\left(\frac{d(a,b)}{\sigma}\right)+g_2 N\left(\frac{-d(a,b)}{\sigma}\right) \tag{4-8}$$

式中 $d(a,b)$为从(a,b)到直线边缘的垂直距离，而 N (•)表示标准正态分布函数。

对于式 (4-7) 和 (4-8)，若固定 $y=b$，考虑水平方向，则有

$$g_{y=b}(a)=g(a,b)=g_1 N\left(\frac{e(b)-a}{\sigma_x}\right)+g_2 N\left(\frac{a-e(b)}{\sigma_x}\right) \tag{4-9}$$

式中 $e(b)$-a 为 $(a，b)$ 到直线边缘的水平距离。

同理，若对式 (4-7) 和 (4-8)，固定 $x=a$，考虑垂直方向，则有

$$g_{x=a}(b)=g(a,b)=g_1 N\left(\frac{e^{-1}(a)-b}{\sigma_y}\right)+g_2 N\left(\frac{b-e^{-1}(a)}{\sigma_y}\right) \tag{4-10}$$

式中 $e^{-1}(a)$-b 为$(a，b)$点到直线边缘的垂直距离。

式 (4-9) 和 (4-10) 中，$e(b)$和 $e^{-1}(a)$分别为 $y=b$ 和 $x=a$ 处直线边缘上的点，因而精确的求出边缘位置对于确保精度非常重要。

二维高斯分布函数 $G(x,y,\sigma)$关于$(0,0)$点是对称的，式 (4-8)，(4-9) 和 (4-10) 可以重新写成：

$$g(a,b)=g_1 N\left(\frac{d(a,b)}{\sigma}\right)+g_2\left[1-N\left(\frac{d(a,b)}{\sigma}\right)\right]=g_2+\Delta g\, N\left(\frac{d(a,b)}{\sigma}\right) \tag{4-11}$$

式中 $\Delta g=g_1-g_2$

$$g_{y=b}(a)=g(a,b)=g_2+\Delta g N\left(\frac{e(b)-a}{\sigma_x}\right) \tag{4-12}$$

$$g_{x=a}(b)=g(a,b)=g_2+\Delta g N\left(\frac{e^{-1}(a)-b}{\sigma_y}\right) \tag{4-13}$$

由式 (4-11)、(4-12) 和 (4-13) 可以得到

$$\frac{\sigma}{d(a,b)}=\frac{\sigma_x}{e(b)-a}=\frac{\sigma_y}{e^{-1}(a)-b} \tag{4-14}$$

又由图 4-2 可以得到：

$$d(a,b)=\frac{|e(b)-a|\cdot|e^{-1}(a)-b|}{\sqrt{(e(b)-a)^2+(e^{-1}(a)-b)^2}} \tag{4-15}$$

由式 (4-14) 和 (4-15) 可以得到σ，σ_x，σ_y之间的关系

$$\sigma=\frac{\sigma_x\cdot\sigma_y}{\sqrt{\sigma_x^2+\sigma_y^2}} \tag{4-16}$$

因此，一个二维问题式 (4-8) 可以分解为两个一维问题式 (4-9) 和 (4-10)。

由式 (4-8) 可知，当已知 g_1、g_2、和某点到边缘的距离 $d(a, b)$和扩展函数σ就可以确定点(a, b)的灰度值 $g(a, b)$。然而，需要求的是相反的问题，即已知 $d(a, b)$和 $g(a, b)$求取σ和 g_1，g_2。如果测试三个点$(a_i, b_i)(i=1,2,3)$，且这三个点到直线边缘的距离都不等，则由式 (4-8) 可以求出 g_1，g_2和σ。

该算法的优点是，σ的求取只用到了图像中距离和灰度信息，没有用微分信息，减少了噪声对其影响，且局部直线边缘可以为任何方向。

为得到精确的深度信息，必须首先得到局部直线边缘的精确位置。如梯度算子、二阶微分算子和形态滤波算子等。二阶微分算子计算图像的二阶微分，把微分值的过零点作为边缘的位置。用二阶微分算子所检测的边缘可以达到像素级灰度。本文使用 Marr 和 Hildreth 提出的 LOG 算子（laplacian-of-Gaussian）。算子的数学表达式为

$$\nabla^2 G(x,y)=\frac{1}{2\pi\rho^4}\left(2-\frac{x^2+y^2}{\rho^2}\right)\exp\left|-\frac{x^2+y^2}{2\rho^2}\right| \tag{4-17}$$

其中 ρ 为常数，实际上使用的是空间滤波器。

4.1.3 深度修正

根据边缘的散焦情况来求解物点深度，即由σ求取 u。其精度不仅与算法有关，还取决于摄像机的光学系统特性。由于像差、镜片的非线性和光学系统的装配误差，都会使由σ估计 u 产生较大的误差。由于物点在坐标系 x-y 坐标不同，使得其像点在图像平面上的坐标不同，根据前面算出的σ值有一定的差异。也即，当图像平面上的边缘位置坐标不同时，即使有相同的σ值，实际物体的深度也会有所不同。因而用σ估计 u 的计算中，应加上由于图像平面上边缘位置的不同的修正量。我们建立三维的查找表，边缘坐标 X、Y 和扩展参数σ为表格的入口，深度修正值 Δu 为求取值，对于表格中不存在的项，用查值算法来得到。

综合以上所述可知，由散焦图像求取物点深度应按以下步骤进行：

1. 首先对系统的 s 和 f 进行标定。标定方法为：取物点在三个不同位置的散焦图像为一组数据，根据式 (4-3) 建立方程组计算对应的 s 和 f 值，并在此基础上建立深度修正量查找表。
2. 对需要获取深度的图像用 LOG 算子找出边缘位置。
3. 在边缘一侧附近找出三个距边缘距离不等的点，并读取这三点的灰度值；
4. 由这三点的灰度值和距边缘的距离通过式 (4-8) 来计算扩展函数 σ；
5. 由扩展函数 σ 来求取深度 u；
6. 该深度 u 加上查找表得到的补偿量为最终的估计深度。

4.2 物体三维运动参数的估计

近年来已有不少人提出利用平面光流估计三维运动参数的方法，Roach & Aggarwal 提出了直接构造非线性方程组求解运动参数的方法，而 Longuet—Higgins 和 Tsai & Huang 提出了线性估计算法。但这些求解方法计算复杂，至少需要已知光流图像上的 8 个点才能唯一确定三维运动参数，而实际上可恢复的运动参数只有 5 个。在已知表面深度的情况下，本文提出了一种快速有效的方法，只需要 5 个光流点就可以唯一确定三维运动参数。

假设摄像机不动，物体在三维空间中运动，则物体的运动由沿摄像机坐标系三个轴的平动和绕三个轴的转动组成。设物体上的点为 $(x, y, z)^t$，旋转角速度矢量为 $\Omega(\omega_x, \omega_y, \omega_z)$，平移速度矢量为 $T(t_x, t_y, t_z)$，物体的运动方程为：

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4-18}$$

摄像机透视投影几何模型见图 4-3，其中 $oxyz$ 为空间坐标系，OXY 为图像坐标系，o 为投影中心，光轴与 z 轴重合，图像平面坐标系的原点位于（0，0，f）处，X 和 Y 轴分别平行于 x 和 y 轴。

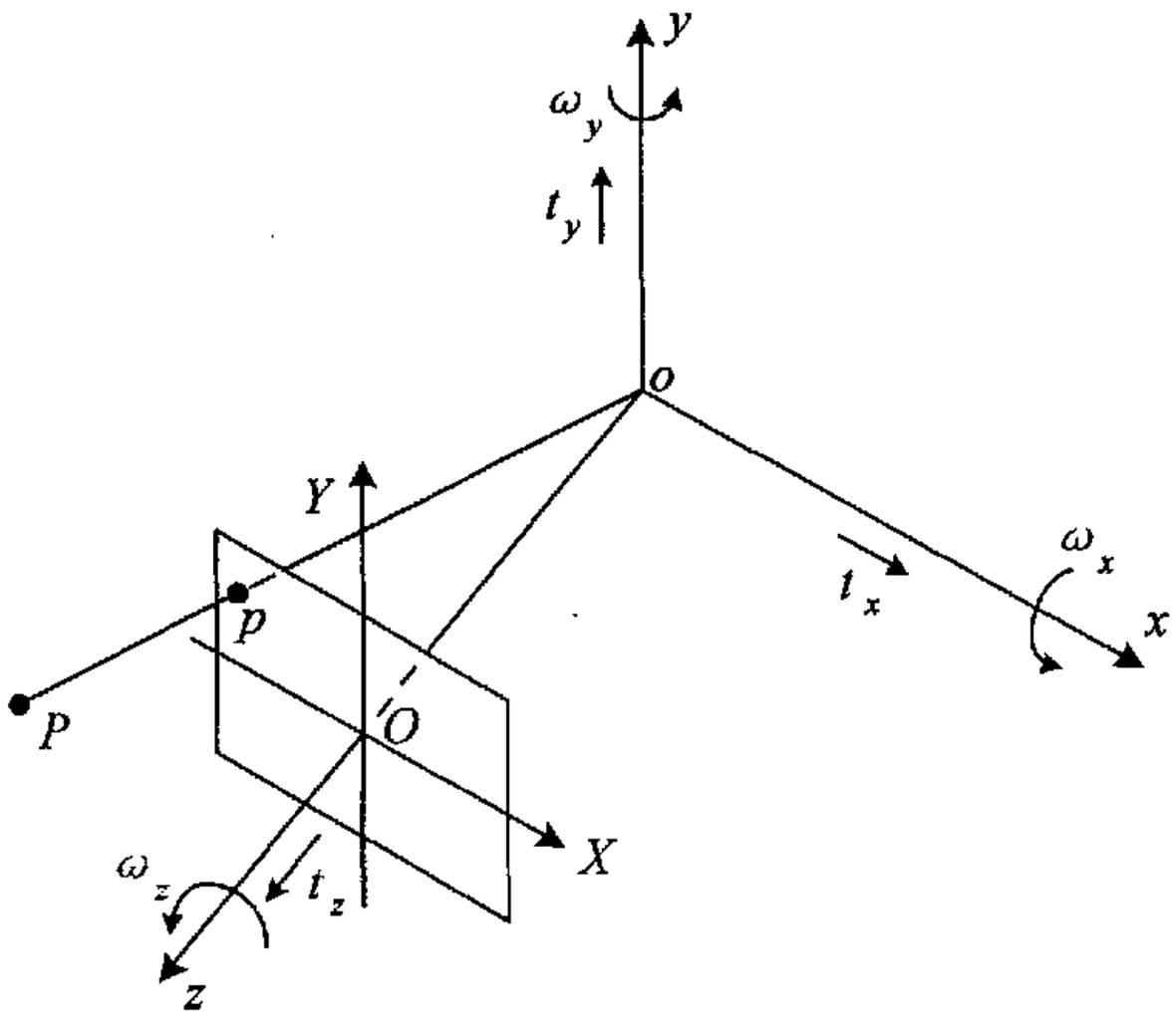


图 4-3 投影几何模型

不失一般性，设摄像机焦距 $f=1$，则投影方程为：

$$\left.\begin{aligned} X &= \frac{x}{z} \\ Y &= \frac{y}{z} \end{aligned}\right\} \tag{4-19}$$

其中 $(X,\ Y)^t$ 为物体上的点 $(x,\ y,\ z)^t$ 在投影面上的投影。

由式 (4-18) 和式 (4-19) 得：

$$\dot{z}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} + z\begin{bmatrix} u \\ v \\ 0 \end{bmatrix} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{4-20}$$

经整理，上式为：

$$\begin{bmatrix} u \\ v \\ 0 \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} + \frac{1}{z}\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} - \frac{\dot{z}}{z}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{4-21}$$

展开式 (4-21)，消去 $\dot{z}$ 并整理得：

$$u = \frac{t_x - Xt_z}{z} + [-XY\omega_x + (1+X^2)\omega_y - Y\omega_z] \tag{4-22}$$

$$v = \frac{t_y - Yt_z}{z} + [-(1+Y^2)\omega_x + XY\omega_y + X\omega_z] \tag{4-23}$$

其中$(u, v)'$为图像点$(X, Y)'$的速度，z为表面深度，利用前一节所述方法可以求得。

从 (4-22) 和 (4-23) 中消去 t_3，得：

$$Yt_x - Xt_y + zX\omega_x + zY\omega_y - (X^2 + Y^2)z\omega_z = uzY - vzX \tag{4-24}$$

要求解的物体三维运动参数有 6 个：t_x、t_y、t_z、ω_x、ω_y和ω_z。从一个光流点可以得到一个光流运动方程，式 (4-24) 中包含了 5 个未知参数 t_x、t_y、ω_x、ω_y和ω_z，因此，利用 5 个光流点建立联立方程组，可求得上述这 5 个未知参数。再把结果代入式 (4-22) 或式 (4-23) 中即可求得 t_3。

4.3 本章小结

1. 用散焦法求取深度实现了用单目摄像机的一幅图像得到物体的三维信息，避免了双目视觉方法的特征匹配的问题，并且计算速度快，能满足实时跟踪的要求，同时采用 4.1.3 节的深度修正法，可获得较高的精度。

2. 在获取物体表面深度的条件下，本文采用一种从光流场恢复物体三维运动参数的强壮算法，只需要 5 个光流点就可以唯一确定物体三维运动参数。

第五章 程序设计及实验

本文采用 Visual C++6.0 编程来实现序列图像分析，估计物体的三维运动参数和结构参数。为了验证本文所述方法的正确性，进行了一系列的实验。

5.1 实验环境的建立

1. 摄像机和采集卡

本课题实验采用德国 VIDEV 生产的 512*582 像素的 CCD 面型摄像机；采用北京嘉恒中自图像技术有限公司生产的 OK 系列视频捕获卡 OK_C20，该卡是基于 PCI 总线的彩色图像采集卡，适用于图像处理、工业监控和多媒体的压缩、处理等研究开发和工程应用领域。可采集单帧，间隔几帧，连续相邻帧的图像，精确到帧，且亮度、对比度、色度等软件可调。

2. 目标体和环境模型

在一块具有光洁平面的金属板上拧上端面边界为正六边形的螺栓，作为目标体和环境模型，采用自然光源。实验平台为三维工作平台。

3. 系统运行环境

在 Windows 98 平台上采用 Visual C++6.0 编程实现，并采用 Matlab6.1 进行辅助模拟仿真。

5.2 视频图像的实时采集

本文采用如图 5-1 所示的流程图进行视频图像采集。在预览回调函数中实现数字图像预处理、稀疏光流场计算以及三维重构，回调函数的声明应放在其调用之前。以下为预览回调函数的声明过程：

```
LRESULT FAR PASCAL FrameCallbackProc (HWND hWnd, LPVIDEOHDR lpVHdr)
{  DWORD dwSize = capGetVideoFormatSize(hWnd); //获取 BITMAPINFO 结构的大小
   lpbii=(LPBITMAPINFO) malloc(dwSize);     //分配空间
   width=lpbii->bmiHeader.biWidth;          //位图的宽度
```

```
    height=lpbii->bmiHeader.biHeight;          //位图的高度
    psize=lpbii->bmiHeader.biBitCount;         //每个像素所用的比特数
    lpBuf=lpVHdr->lpData;                      //返回图像数据在内存中的首地
址
      .....
  }
```

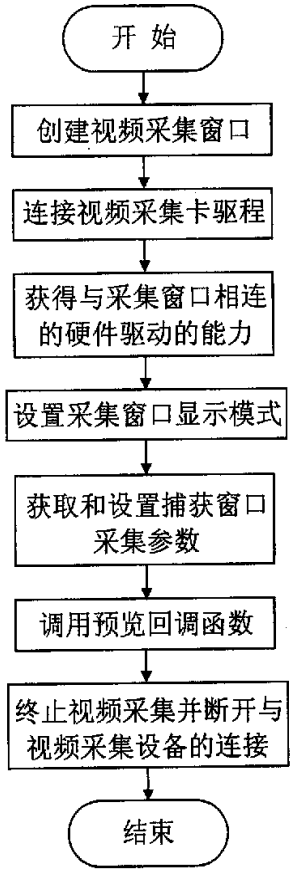


图 5-1 视频图像采集流程图

程序运行界面如图 5-2 所示：

图 5-2 程序运行界面

选择图像采集按钮后，会弹出视频源设置对话框，如图 5-3 所示：

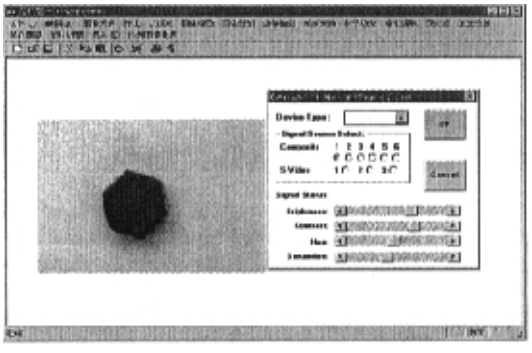


图 5-3 视频源设置对话框

设置完视频源对话框，单击 OK 按钮，会出现如图 5-4 所示的视频格式设置对话框，本论文中对 RGB 8bit、RGB 24bit 和 RGB 24bit 都进行了编程，选用的是 320*240 的采集窗口。

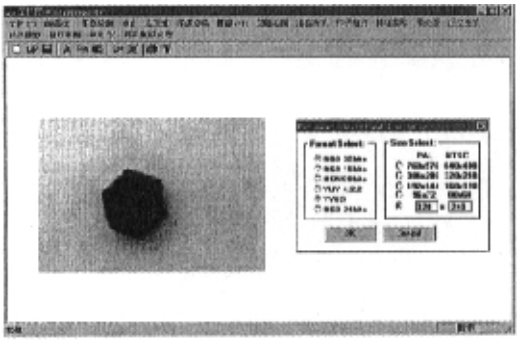


图 5-4 视频格式设置对话框

5.3 数字图像预处理

1. 内存块的分配

图像处理都是在内存中进行的，Win32 提供了 3 种操纵内存的机制：

- 虚拟内存，最适合于管理大的对象数据或结构；
- 内存映射文件，最适合于管理大的数据流（通常来自文件）和在多个进程间共享数据；
- 堆，最适合于管理大量的小对象。

Win32API 具有向前兼容性，可以使用 16 位 Windows 堆函数,因此在 Win32API 中可以使用 32 位和 16 位两种堆函数。但使用 16 位 Windows 堆函数时，需要分配额外的一块内存来存放句柄表项，因此它同 32 位的 Windows 堆函数相比，执行缓慢，需要更多的内存。

本文图像文件数据流较小，且需要分配很多内存块，因此适合采用堆来进行内存分配，本文采用 32 位的 Windows 堆函数实现了真正的 32 位内存分配。具体实现如下：

```
HANDLE GetprocessHeap(VOID); //获得进程的缺省堆句柄
LPVOID HeapAlloc(HANDLE hHeap, DWORD dwFlags, DWORD dwBytes);
                                    // 从堆中分配内存，分配成功，则返回块的地址
DWORD HeapSize(HANDLE hHeap, DWORD dwFlags, LPCVOID lpMem);
    // 内存块分配成功后，可以调用 HeapSize 函数来得到该块的实际大小；
BOOL HeapFree(HANDLE hHeap, DWRD dwFlags, LPVOID lpMem);
    // 当不再需要内存块时，可以调用 HeapFree 来释放它
BOOL HeapDestroy(HANDLE hHeap);        // 释放堆
```

2. 实验

对采集的图像进行预处理，主要是为了去除图像中噪声的干扰，增强图像中的有用信息，以便容易提取出目标物体的边缘，并且通过种子填充把图像中目标体边缘内外的像素点区分开来，为下一步特征点的提取和序列图像间特征点的匹配做好准备。具体的预处理过程流程图如图 5-5 所示。

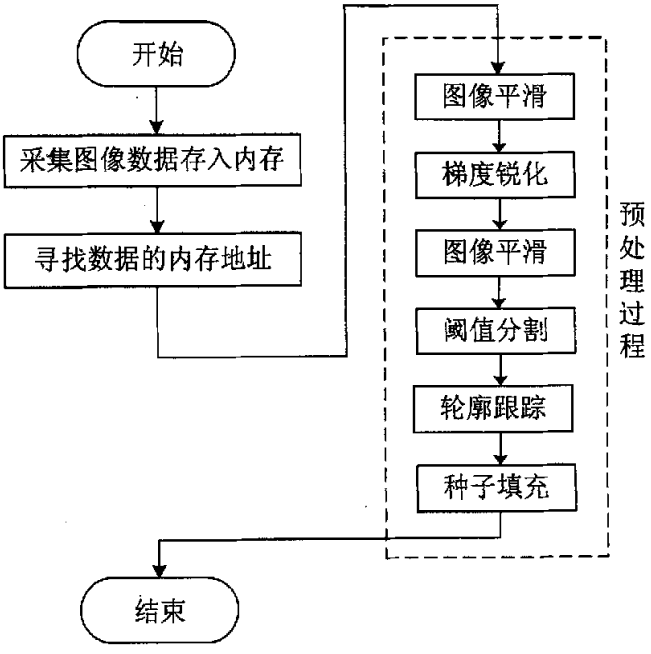


图 5-5 预处理过程流程图

图像处理程序运行界面如图 5-6 所示，其中左边为处理前的图像，右边为处理后的图像。

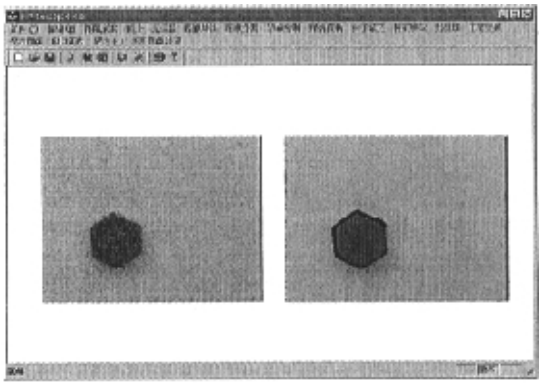
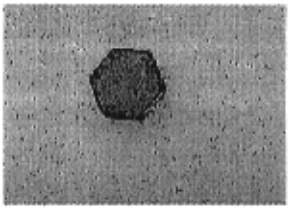
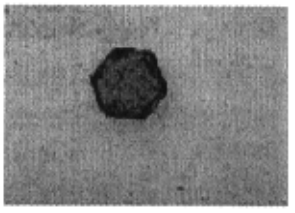


图 5-6 图像处理程序运行界面

图 5-7 给出了采用本文所述算法对图像进行平滑处理前后的比较图，其中(a)为处理前的原图，(b)为平滑处理后的图像。从图中可以看到，处理后图像的噪声较处理前有明显的减少，起到了平滑处理的效果。



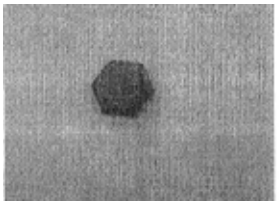
(a)图像平滑前



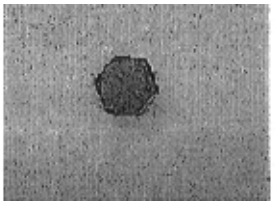
(b)图像平滑后

图 5-7 平滑处理前后的图像

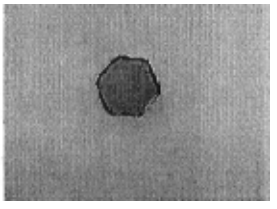
图 5-8 比较了梯度锐化前后的图像，其中(a)为采集的原图像，(b)为未经平滑处理直接进行梯度锐化后的图像，(c)为经过平滑再进行梯度锐化后的图像。



(a)原图像



(b)未经平滑处理
梯度锐化图

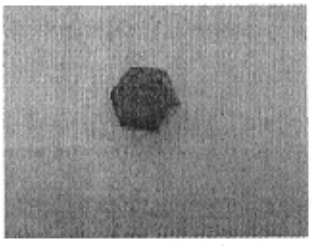


(c) 经过平滑处理
梯度锐化图

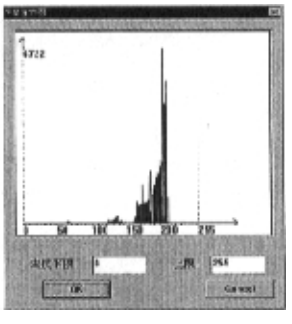
图 5-8 梯度锐化前后的图像

从图中可以看出，对图像进行梯度锐化可以增强图像的边缘、细节等有用信息，但同时增大了图像的噪声，因此，在进行梯度锐化前，必须先对图像进行平滑处理，以去除图像中的噪声干扰，使梯度锐化后的图像变得比较清晰。

图5-9为原图像及其灰度直方图，图5-10为梯度锐化后图像及其灰度直方图。从灰度直方图可以看出，梯度锐化后图像灰度变得比较集中，物体的边缘得到了增强。

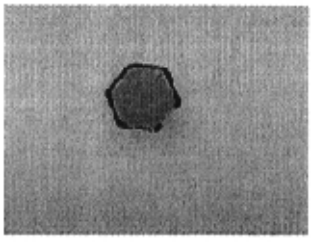


处理前的原图

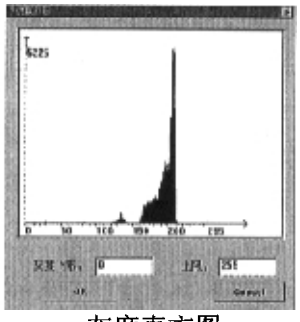


灰度直方图

图 5-9 原图像及其灰度直方图



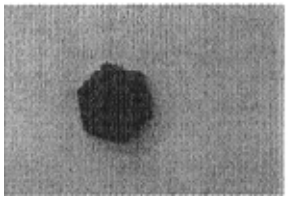
梯度锐化后



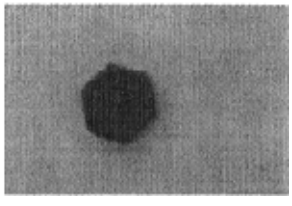
灰度直方图

图 5-10 梯度锐化后图像及其灰度直方图

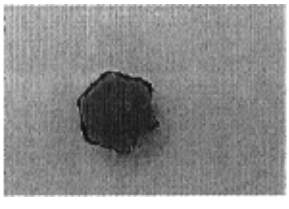
图 5-11 给出了对摄取的图像进行预处理的全过程图。



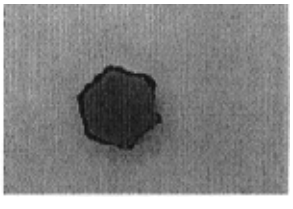
摄取图像



平滑处理



梯度锐化



平滑处理

阈值分割

边缘提取

种子填充

图 5-11 图像预处理过程图

5.4 稀疏光流场计算

稀疏光流场计算分三步进行，首先分别提取两帧图像的特征点，其次进行两帧图像间特征点的匹配，最后再进行特征点处稀疏光流的计算。流程图如图 5-12 所示。

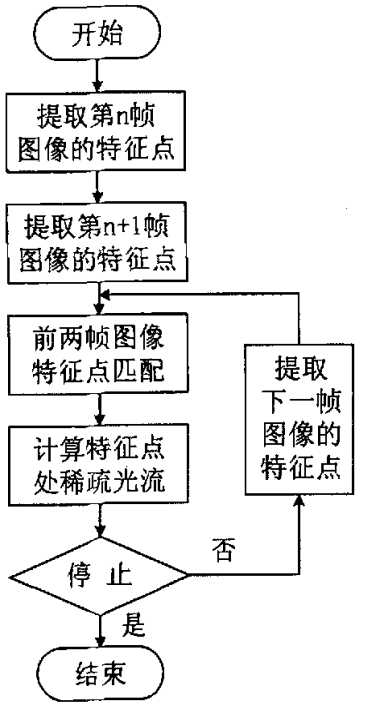


图 5-12 稀疏光流场计算流程图

1. 特征点提取实验

为了验证 3.2.1 节所述方法的正确性，进行了如下实验。图 5-13 为预处理后的图像，图 5-14 为利用 3.2.1 节所述方法提取出的角点图。

图 5-13 预处理后图像

图 5-14 提取特征点后

2. 用 Hopfield 神经网络实现松弛匹配实验

本文取了多组特征点来进行序列图像间特征点匹配实验。首先利用式(3-31)确定 Hopfield 神经网络的初始状态，利用式(3-30)和式(3-29)分别确定网络的连接权矩阵和输入偏置，并利用式(3-15)和式(3-16)进行网络的演化。当网络的能量变化低于某个预先设定的门限以后，就可认为系统已经收敛。图 5-15 给出了待匹配的一组特征点，表 5-1 为图 5-15 (a)中特征点的坐标值，表 5-2 为图 5-15 (b)中特征点的坐标值，表 5-3 为正确匹配时(b)在(a)中的特征对应点，表 5-4 为采用本文所述方法进行匹配时(b)在(a)中的特征对应点。

(a) (b)

图 5-15 待匹配的一组特征点

表 5-1 (a)中特征点的坐标

序号	坐标值	序号	坐标值
1	(142, 141)	4	(60, 105)
2	(108, 166)	5	(95, 78)
3	(64, 147)	6	(135, 94)

表 5-2 (b)中特征点的坐标

序号	坐标值	序号	坐标值
1	(132, 76)	4	(54, 141)
2	(131, 150)	5	(57, 78)
3	(93, 169)	6	(91, 81)

表 5-3 正确匹配时的特征对应点

(a)中特征点序号	1	2	3	4	5	6
(b)中特征点序号	2	3	4	5	6	1

表 5-4 采用本文方法时的特征对应点

(a)中特征点序号	1	2	3	4	5	6
(b)中特征点序号	2	3	4	5	6	1

3. 稀疏光流场计算实验

利用 3.2 节所述方法提取出特征点并在序列图像间进行匹配之后，利用视差就可以计算特征点处的光流。图 5-16 为利用序列图像计算光流示意图，图 5-17

为本文实验所得的几种光流形式。

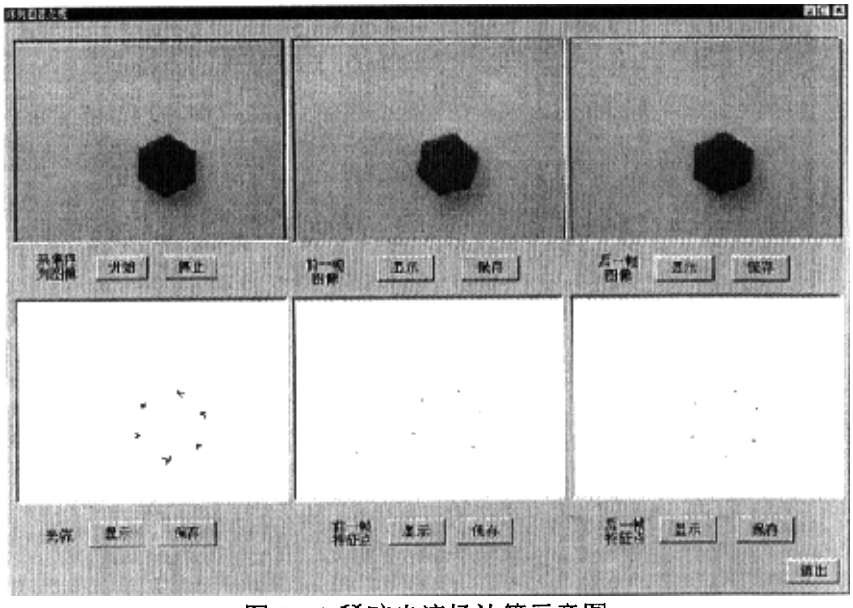
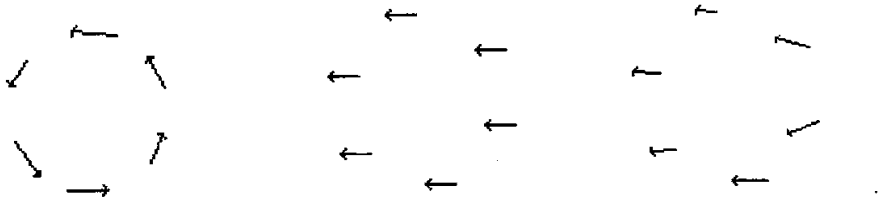


图 5-16 稀疏光流场计算示意图



(a) 纯转动光流　　(b) 纯平动光流　　(c) 一般运动光流

图 5-17 几种光流形式

5.5 散焦法估计物体深度实验

为了验证利用散焦法估计物体深度方法的正确性，进行了如下实验。图 5-18 为聚焦图像，图 5-19 为物体在某一深度距离下的散焦图像。

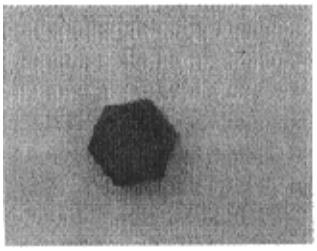


图 5-18 聚焦图像

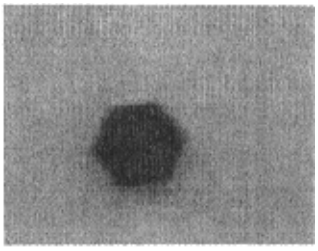


图 5-19 散焦图像

表 5-5 列出了数次实验结果。从实验结果可以看出，采用本文所述方法进行深度估计时，其误差小于 5%，能够满足一般的使用要求。

表 5-5 用散焦法测深度实验结果

实验次数	物体的实际深度(mm)	物体的估计深度(mm)
1	40	42
2	45	47
3	50	48
4	55	57
5	60	59
6	65	62
7	70	70
8	75	76
9	80	83
10	85	86

5.6 本章小结

采用 Visual C++6.0 语言编程来实现序列图像分析，执行效率高，能满足实时跟踪的要求。本章在对程序进行介绍的基础上，做了大量实验来验证本文所述方法的正确性。

第六章 结论与展望

由于序列图像分析在国民经济和军事领域的许多方面有着广泛的应用，因此对它的研究受到各国科学家的普遍关注，近年来发展的研究方法主要有两种：基于特征的方法和基于光流的方法。本文综合两种方法的优点，提出了利用稀疏光流进行序列图像分析，估计物体三维运动参数和结构参数的方法，并进行了相关实验。现将其中主要的具有创造性的研究成果归纳如下：

1. 使用 VC++提供的 VFW (Video For Windows) 函数库，开发基于 Windows 的视频捕获应用程序，提高了程序的通用性；
2. 本文采用单目视觉方法，避免了多摄像机的的匹配问题，提高了系统的稳定性和识别速度。采用散焦法估计深度，并进行深度修正以满足测量精度要求，从而解决了单目视觉估计深度的难题。
3. 本文对序列图像进行了图像平滑、阈值分割和边缘跟踪等一系列预处理，去除了图像中的噪声，提取出了目标物体的边缘点。采用 32 位的 Windows 堆函数在内存中对图像数据进行处理，实现了真正的 32 位内存操作。
4. 采用面积约束和夹角约束相配合的方法提取特征点，运算量小，计算速度快；采用 Hopfield 神经网络实现序列图像间特征点的松弛匹配，把松弛匹配的准确性和 Hopfield 神经网络强大的并行处理能力结合起来，快速而有效地实现了序列图像间特征点的匹配。
5. 考虑到估计物体运动所需特征点数量的要求，本文采用稀疏光流法进行序列图像分析，克服了经典光流法计算速度慢，对噪声敏感等缺点，能够更好地满足实时跟踪的要求。

由于实验室条件和开发时间的限制，本文开发的系统在以下几个方面还有待于进一步完善：

1. 利用稀疏光流法得到的是物体特征点处的光流，要进行三维表面重建等后续研究，还需要利用插值得到图像中每个像素点的光流；
2. 由于光照条件对图像处理精度有很大影响，在光照条件比较恶劣的情况下，还有待于发展更为有效的算法来提高系统运行的稳定性和准确性。

参考文献

[1] 李智勇，沈振康，动态图像分析，北京：国防工业出版社，1999

[2] Ullman S., The Interpretation of Visual Motion, MIT Press, 1979

[3] X. Zhuang, T.S. Huang, R.M. Haralick, A simple procedure to solve motion and structure from three orthographic view, IEEE J. Robotics and Automation, 1988,4(2)

[4] T.S. Huang, C.H. Lee, Motion and structure from orthographic projections, IEEE Trans. Pattern Anal. Machine Intell., vol.11, no.5, May, 1989.536-540

[5] J.W. Roach and J.K. Aggarwal, Determining the movement of objects from a sequence of images, IEEE Trans. Pattern Anal. Machine Intell., vol.PAMI-2, no.6, Nov., 1980.554-562

[6] R.Y. Tsai, T.S. Huang, Uniqueness and estimation of 3-D motion parameters of rigid bodies with curved surfaces, IEEE Trans. Pattern Anal. Mach. Intell., 1984, 6(1)

[7] Longuet – Higgins H.C., A computer algorithm for reconstructing a scene from two projections, Nature, vol. 293, no.5828, Sept., 1981.133-135

[8] Longuet – Higgins H.C., The reconstruction of a scene from two projections configurations that defeat the 8 – point algorithm, Dencer Colorado: Proc. 1st Conf. Artif. Intell. Application, 1984

[9] Zhuang X, Huang T.S., Haralick R.M. Two – view motion analysis: a unified algorithm, J. Opt. Soc. Am., 1986, A3(9)

[10] 高满屯，从序列图像获取运动刚体的三维信息，工程图形学报，1989，11(1)

[11] B.L. Yen and T.S. Huang, Determining 3-D motion and structure of a rigid body using straight line correspondences, IEEE International Conference on Acoustics, Speech and Signal Processing, vol.1, 1983.118-121

[12] B.K.P. Horn and B.G. Schunck, Determining optical flow, Artificial Intelligence, vol.17, 1981.185-203

[13] B.Lucas and T.Kanade, An iterative image registration technique with an application to stereo vision, Proc. DARPA Image Understanding Workshop, 1981.121-130

[14] D.Terzopoulos, Regularization of inverse visual problems involving discontinue-tyies, IEEE Transaction on Pattern Analysis and Machine Intelligence, 8(4), 1986. 413-424

[15] E.C.Hildreth, The Measurement of Visual Motion, Cambridge, Massachusetts: MIT Press, 1984

[16] M.Yachida, Determining velocity map by 3 – D iterative estimation, Proceedings of the 7th International Joint Conference on Artificial Intelligence, 1981.716-718

[17] R.M.Haralick and J.S.Lee, The facet approach to optical flow, Proc. Image Understanding Workshop, Arlington, VA, 1984.74-83.

[18] O.Tretiak and L.Pastor, Velocity estimation from image sequences with second order differential operators, Proc. 7th International Conference on Pattern Recognition, Montreal, Canada, 1984.16-19

[19] H.H.Nagel, On the estimation of optical flow: relation between different approaches and some new results, Artificial Intelligence, 33(3), 1987.229-324

[20] 傅洁和吴立德, 计算机视觉中运动分析的连续处理方法综述[J],模式识别与人工智能，1991，4(1)：91-100

[21] E.H.Adelson and J.R.Bergen, Spatiotemporal energy models for the perception of motion, Journal Optic Society of American, A2, 1985.284-229

[22] P.Anandan, A computational framework and an algorithm for the measurement of visual motion, International Journal of Computer Vision, 2(3), 1989.283-310

[23] S.S.Beauchemin and J.L.Barron, The computation of optical flow, ACM Computing Surveys, 27(3), 1995.433-467

[24] A.Pentland, New sense for depth of field, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.PAMI-9, no.4, July, 1987.523-531

[25] M.Subbarao and N.Gurumoorthy, Depth recovering from blurred edges, IEEE CVPR, 1988

[26] Shang – Hong Lai and Chang – Wu Fu, A generalized depth estimation algorithm with a single image, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.14, no.4, April, 1992.407-411

[27] Thomas S.Huang, Motion and structure from feature correspondences: A Review, Proceeding of the IEEE, vol.82, no.2, February 1994

[28] 焦李成，神经网络计算，西安电子科技大学出版社，1996

[29] Hongche Liu, Accuracy vs. efficiency trade-offs in optical flow algorithms, Computer Vision and Image Understanding, vol.72, no.3, December, 1998.271-286

[30] A.Goshtasby, G.Stockman, Point pattern matching using convex hull edges, IEEE Trans SMC., vol.15, no.5, 1985.631-637

[31] H.Ogawa, Labeled point pattern matching by Delaunay triangulation and maximal cliques, Pattern Recognition, vol.19, no.1, 1986.35-40

[32] S.Ranade, A.Rosenfeld, Point pattern matching by relaxing, Pattern Recognition, vol.12, no.4, 1980.269-275

攻读硕士学位期间发表的论文

1. Wanggang, Zhangshufang, "Spatial Trajectory Prediction of Visual Servoing", Chinese Journal of Mechanical Engineering, vol.16, no.1, March, 2003.7-9

2. 李建军，郭玉申，王刚，张淑芳，基于单片机的语音系统通用开发方法研究，机床与液压，已录用。

致　　谢

本论文是在导师王刚副教授的悉心指导下完成的。王刚导师严谨的治学态度、渊博的知识以及极强的科研能力令我钦佩不已，是我学习的榜样。研究生期间，王刚导师在生活上和科研上给了我极大的帮助，导师的言传身教使我收益匪浅。在此向尊敬的导师表示由衷的感谢和崇高的敬意。

在攻读硕士学位期间，还得到了阎祥安教授、洪鹰副教授、赵臣副教授、王国栋副教授、郭玉申副教授在学习和生活上的指导和帮助，在此向他们表示衷心的感谢。

在撰写论文期间，崔润龙、吴雪艳、胡明艳、黄晶、张宏伟、李建军、宁文军、刘科、任国丽等同学给予了热情帮助，在此向他们表达我的感激之情。

另外也感谢家人对我的理解和支持，使我能够在学校专心完成我的学业。