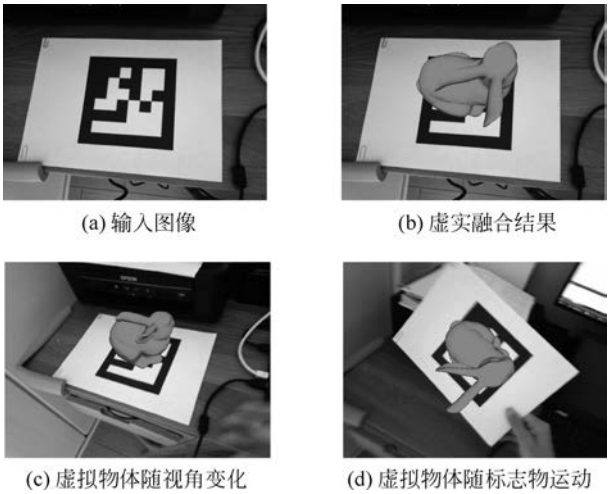
# 第3章 基于平面标志的空间注册

CHAPTER 3

在增强现实中，标志指现实场景中存在的易于识别和定位的物体，其主要作用是辅助相机或者物体的定位跟踪，以较为简单、稳定的方式实现空间注册。在现有增强现实系统中，平面标志是应用最为广泛的一种标志技术，这主要是因为平面的制作、表示和计算都较为简单。平面标志既可以由人工设计制作，也可以是场景中自然存在的平面物体（如书页、海报等）。本章将以人工平面标志为基础，实现一个最为基本的增强现实系统。如无特殊说明，本章所述平面标志皆为人工平面标志，对自然标志物的处理方法将在第5章进行介绍。

## 3.1 系统概述

如图3.1所示为本章将要介绍的简易增强现实系统。其中图3.1(a)是从视频捕获的输入帧，其中包含一个平面标志；图3.1(b)是在图3.1(a)的基础上绘制了虚拟物体之后的增强现实效果，注意虚拟物体看起来就像被放在了平面标志上一样；图3.1(c)和图3.1(d)是视频中另外两个时刻的效果图，可以看到，当摄像机运动或者平面标志移动的时候，虚拟物体都会随着平面标志一起运动，看起来像是真的固定在平面标志上的物体一样。这就是本书第1、2章讲到的虚实空间注册所要追求的效果。在这里，即是将虚拟物体所在的局部空间与根据平面标志定义的现实空间进行了注册。



(a) 输入图像　(b) 虚实融合结果

(c) 虚拟物体随视角变化　(d) 虚拟物体随标志物运动

图3.1　基于平面标志的增强现实系统

视频演示

为了实现上述虚实融合效果，就需要知道绘制虚拟物体所需的参数。根据本书第1、2章的内容和计算机图形学的相关知识可知，所需的参数主要包含两类：一是虚拟物体相对于相机的位置和姿态参数，可以表示为虚拟物体从其局部坐标系变换到相机坐标系的旋转矩阵 $\boldsymbol{R}$ 和平移向量 $\boldsymbol{t}$，根据这个参数可以将虚拟物体嵌入相机坐标系中的对应位置；二是相机的内参矩阵 $\boldsymbol{K}$，用于确定相机坐标系中一个3D空间点在图像上的对应像素点位置。在OpenGL绘制程序中，$\boldsymbol{R}$ 和 $\boldsymbol{t}$ 将用于决定模型-视图变换，而 $\boldsymbol{K}$ 将决定投影变换。关于参数 $\boldsymbol{R}$、$\boldsymbol{t}$、$\boldsymbol{K}$ 与OpenGL中各变换矩阵的转换关系，将在3.4节进行介绍。

对于定焦相机而言，相机的内参矩阵 $\boldsymbol{K}$ 是固定不变的，并通过2.4.1节介绍的相机标定方法获得。因此，关键是计算外部参数 $\boldsymbol{R}$ 和 $\boldsymbol{t}$。由于 $\boldsymbol{R}$ 和 $\boldsymbol{t}$ 跟摄像机位置和虚拟物体在场景中的摆放位姿相关，因此需要根据输入视频流进行实时计算。这也是所有增强现实系统需要解决的首要问题。在一般情况下，这需要持续稳定地跟踪摄像机的3D位姿参数，同时还要知道虚拟物体摆放处的部分现实场景的3D几何，二者皆是非常具有挑战性的问题。

在本章中，将主要介绍基于平面标志的 $\boldsymbol{R}$ 和 $\boldsymbol{t}$ 的简化计算方法。为此，首先假设算法的目标是把虚拟物体放置到平面标志上，如图3.1所示的效果。这样，虚拟物体在相机坐标系中的位姿，实际上也就是平面标志在相机坐标系中的位姿；或者更一般地，二者之间只差一个不随时间变化的恒定变换，可以在系统初始化的时候设定，相当于指定虚拟物体和平面标志之间的相对位姿关系。只要知道了平面标志相对于摄像机的位姿参数，就可以容易地得到虚拟物体的位姿参数 $\boldsymbol{R}$ 和 $\boldsymbol{t}$。

## 3.2　平面位姿估计

计算平面标志在相机坐标系中的位姿的基本方法是，首先获得一组平面标志上的3D空间点与图像上相应像素点的对应关系，即3D-2D点对，再以此为约束求解位姿参数。实际上，这也是求解一般3D物体位姿的基本方法。对于一般3D物体而言，根据一组3D-2D点对求解其姿态参数可以采用PnP(Perspective-n-Point)方法，相关内容将在4.2节中进行介绍。对于平面物体的位姿而言，虽然也可以采用PnP方法进行计算，但是由于平面的3D位姿存在特殊表示，即单应性矩阵，因此在本节中，先介绍一种专门针对平面的位姿计算方法。这里假设已知一组3D-2D点对，如何获取这些点对的方法将在3.3节中介绍。

### 3.2.1　单应性矩阵

回顾2.3节介绍的针孔相机模型，3D空间点 $\boldsymbol{X}$ 投影到图像上相应像素点 $\boldsymbol{x}$ 的过程可以表示为

$$\tilde{\boldsymbol{x}} \sim \boldsymbol{K}(\boldsymbol{R} \quad \boldsymbol{t})\tilde{\boldsymbol{X}} \tag{3.1}$$

式中，变量上方的符号“～”表示相应的齐次坐标，连接两组代数式的符号“～”则表示其中一边乘以一个恰当的数字会使两边相等。对于平面物体而言，可以为其选取一个局部坐标系，使得该平面物体上任意一个3D空间点的 $Z$ 坐标都等于0，因此有

$$\tilde{\boldsymbol{x}} \sim \boldsymbol{K}(\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3 \quad \boldsymbol{t})\begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \boldsymbol{K}(\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t})\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \boldsymbol{H}\begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} \tag{3.2}$$

其中

$$\boldsymbol{H} \sim \boldsymbol{K}(\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t}) \tag{3.3}$$

是一个 3×3 的可逆矩阵，$\boldsymbol{r}_1$、$\boldsymbol{r}_2$、$\boldsymbol{r}_3$ 是旋转矩阵 $\boldsymbol{R}$ 的列向量。

上式中的矩阵 $\boldsymbol{H}$ 不仅包含物体的旋转和平移信息，同时还与相机内部参数 $\boldsymbol{K}$ 有关，可以表示 3D 场景中的平面物体通过透视变换映射到图像平面的过程，称其为 3D 平面的**单应性变换**，相应的变换矩阵被称为**单应性矩阵**。单应性变换其实是一个 2D 的射影变换，其基本特点是保持直线性，即直线经单应性变换之后仍然为直线。一个单应性变换的典型例子是如图 3.2 所示的中心投影。$\pi$ 和 $\pi'$ 是 3D 空间的两个平面，$O$ 是空间中一点，并且不在这两个平面上。这样，平面 $\pi$ 上任意一点 $\boldsymbol{x}$ 通过与 $O$ 的连线都与平面 $\pi'$ 相交，且交点唯一，即 $\boldsymbol{x}'$。这样就建立了从平面 $\pi$ 到 $\pi'$ 的一个一一映射，且这个映射显然是可逆并且保持直线性的，因此是一个单应性变换，可以用单应性矩阵表示。

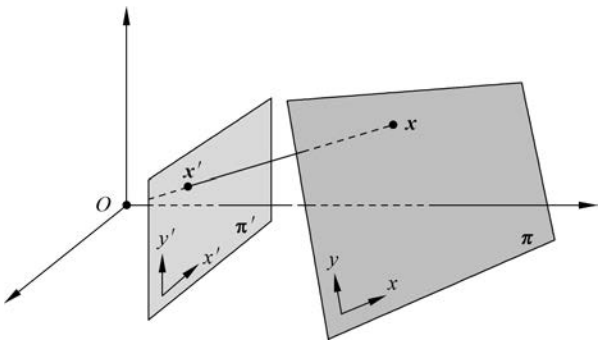



图 3.2　中心投影

实际上，上述中心投影模型与针孔相机的成像过程也是一致的，中心点 $O$ 相当于相机的光心，若平面 $\pi'$ 与光轴正交，则相当于相机的成像面。因此，平面 $\pi$ 到平面 $\pi'$ 的映射总是可以用一个单应性矩阵来表示。这也正是式(3.2)所表示的情况，在本章中将使用单应性矩阵来表示和计算平面标志的 3D 位姿。

根据中心投影模型和单应性变换的性质，可以得出其他一些可以用单应性变换来表示的情况。典型地，如果 $\pi$ 是场景中的一个 3D 平面，则其在不同视点 $O$ 和 $O'$ 下所观测到的图像上的像素对应关系也可以表示为一个单应性变换。实际上，设平面 $\pi$ 上的点 $\boldsymbol{X}$ 通过矩阵 $\boldsymbol{H}_1$ 映射到第一张图像上的点 $\boldsymbol{x}$，即 $\tilde{\boldsymbol{x}} \sim \boldsymbol{H}_1 \widetilde{\boldsymbol{X}}$，并有 $\widetilde{\boldsymbol{X}} \sim \boldsymbol{H}_1^{-1} \tilde{\boldsymbol{x}}$；通过矩阵 $\boldsymbol{H}_2$ 映射到第二张图像上的点 $\boldsymbol{x}'$，即 $\tilde{\boldsymbol{x}}' \sim \boldsymbol{H}_2 \widetilde{\boldsymbol{X}}$，并有 $\widetilde{\boldsymbol{X}} \sim \boldsymbol{H}_2^{-1} \tilde{\boldsymbol{x}}'$，则经过简单的推导，有

$$\bar{\boldsymbol{x}}' \sim \boldsymbol{H}_2 \boldsymbol{H}_1^{-1} \bar{\boldsymbol{x}} = \boldsymbol{H} \bar{\boldsymbol{x}} \tag{3.4}$$

在式(3.4)中，$\boldsymbol{H} = \boldsymbol{H}_2 \boldsymbol{H}_1^{-1}$。显然，矩阵 $\boldsymbol{H}$ 仍然是 3×3 的可逆矩阵，因此也是一个单应性矩阵。进一步地，可以得到，任何两个平面都可以通过一个 3×3 的矩阵建立一一映射。对于不同视频帧中观测到的场景中的同一个平面而言，可以用一个单应性变换来建立两帧之间的像素对应关系，这在图像匹配中有重要作用。

另一个典型的情况是，由旋转相机(相机光心静止，只有绕光心旋转的运动)拍摄到的不同图像之间的像素对应关系，也可以用一个单应性变换来表示。这种情况也可以用如图 3.2 所示的中心投影来解释，此时平面 $\pi$ 和平面 $\pi'$ 相当于相机在不同旋转角度下的对应成像

面，且由于相机光心和场景 3D 点都没有发生变化，只有相机成像面在运动，因此也是符合中心投影模型的。注意，对于旋转相机的情况而言，即使所拍摄的 3D 场景不是平面，也可以用单应性变换来精确表示像素的对应关系。实际上，当视点固定时，基线长度为 0，因此有 $\boldsymbol{t}=\boldsymbol{0}$，坐标变换矩阵为

$$\boldsymbol{T}=\begin{pmatrix}\boldsymbol{R} & \boldsymbol{0}\\ \boldsymbol{0}^{\mathrm{T}} & 1\end{pmatrix}=\begin{pmatrix}r_{11} & r_{12} & r_{13} & 0\\ r_{21} & r_{22} & r_{23} & 0\\ r_{31} & r_{32} & r_{33} & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.5}$$

因此，透视投影矩阵为

$$\boldsymbol{P}=\boldsymbol{K}(\boldsymbol{I}\quad\boldsymbol{0})\begin{pmatrix}\boldsymbol{R} & \boldsymbol{0}\\ \boldsymbol{0}^{\mathrm{T}} & 1\end{pmatrix}=\boldsymbol{K}(\boldsymbol{R}\quad\boldsymbol{0})=(\boldsymbol{KR}\quad\boldsymbol{0}) \tag{3.6}$$

可见，此时的透视投影矩阵也退化为 3×3 的矩阵，也就是说，尽管场景中的物体形状各异，远近不同，但是，摄像机在不同姿态下拍摄的图像之间，采用 3×3 的单应性矩阵就可以表达两幅图像间的映射关系。在使用手机的全景拍照功能时会发现，当相机运动接近一个旋转相机时，全景图的拼接效果最好，其原因就在于，只有旋转相机才能在不恢复场景 3D 几何结构的情况下，基于图像的单应性变换将不同角度拍摄到的图像进行精确对齐。

### 3.2.2　求解单应性矩阵

为对单应性矩阵进行求解，将单应性矩阵表示为如下形式：

$$\boldsymbol{H}=\begin{pmatrix}h_{11} & h_{12} & h_{13}\\ h_{21} & h_{22} & h_{23}\\ h_{31} & h_{32} & h_{33}\end{pmatrix} \tag{3.7}$$

因此，求解单应性矩阵即是要求解式(3.7)中的 9 个矩阵参数，将其表示为一个参数向量 $\boldsymbol{h}=(h_{11}\quad h_{12}\quad h_{13}\quad h_{21}\quad h_{22}\quad h_{23}\quad h_{31}\quad h_{32}\quad h_{33})^{\mathrm{T}}$。注意，由于单应性变换前后都使用齐次坐标，因此，将单应性矩阵乘以一个非零常数，其所表示的变换是不变的。不失一般性，假定 $\boldsymbol{h}$ 是一个归一化向量，即 $\|\boldsymbol{h}\|=1$，那么，虽然一个单应性矩阵包含 9 个未知参数，但其自由度实际上为 8。由于每个点对可以提供两个约束，所以最少需要 4 个平面到平面的点对才能唯一确定 $\boldsymbol{h}$。

如果给定一组平面到平面的点对集合，就可以采用**直接线性变换**(direct linear transform，DLT)来估计满足这组点对约束的最优单应性变换。记$(x_i,y_i)$和$(x'_i,y'_i)$是一个对应点对，则二者应满足以下约束条件：

$$x'_i=\frac{h_{11}x_i+h_{12}y_i+h_{13}}{h_{31}x_i+h_{32}y_i+h_{33}},\quad y'_i=\frac{h_{21}x_i+h_{22}y_i+h_{23}}{h_{31}x_i+h_{32}y_i+h_{33}} \tag{3.8}$$

上述形式对待求解参数 $\boldsymbol{h}$ 是非线性的。不过，可以在式(3.8)两边同时乘以分母，并将式(3.8)直接变换为如下线性形式：

$$\left.\begin{aligned}h_{11}x_i+h_{12}y_i+h_{13}-x'_ix_ih_{31}-x'_iy_ih_{32}-x'_ih_{33}=0\\ h_{21}x_i+h_{22}y_i+h_{23}-y'_ix_ih_{31}-y'_iy_ih_{32}-y'_ih_{33}=0\end{aligned}\right\} \tag{3.9}$$

将所有特征点对通过上述形式进行排列，可以得到：

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 & -x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 & -y'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix} \tag{3.10}$$

式(3.10)中的系数矩阵记为 $\boldsymbol{A}$，则方程具有 $\boldsymbol{Ah}=\boldsymbol{0}$ 的形式，是一个齐次线性方程组，一般情况下可基于奇异值分解(singular value decomposition，SVD)进行求解。具体地，如果矩阵 $\boldsymbol{A}$ 的 SVD 形式为 $\boldsymbol{A}=\boldsymbol{UDV}^{\mathrm{T}}$，且对角阵 $\boldsymbol{D}$ 的对角线元素按降序排列，则待求解结果是矩阵 $\boldsymbol{V}$ 的最后一列。由于 $\boldsymbol{V}$ 是正交矩阵，所以，这样得到的参数向量 $\boldsymbol{h}$ 也满足 $\|\boldsymbol{h}\|=1$ 的约束。

DLT 方法的一个缺点是，在式(3.8)的两边同时乘以分母，相当于给每一个点对关联的方程乘以一个权重，将导致坐标值较大的点对结果有较大的影响，这显然是不合理的。为了获得更精确的结果，往往以 DLT 方法所获得的结果为初值，并进一步采用非线性优化的方法来最小化重投影误差。

在非线性估计方法中，最常采用的是非线性最小二乘法。假设$\{(\boldsymbol{x}_i,\boldsymbol{x}'_i),i=1,2,\cdots\}$是输入点对，$\boldsymbol{f}$ 是从 $\boldsymbol{x}_i$ 到 $\boldsymbol{x}'_i$的变换，即 $\boldsymbol{x}'_i=\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\Theta})$，其中$\boldsymbol{\Theta}$ 是待估计运动模型的参数向量。非线性最小二乘法需要对当前估计参数$\boldsymbol{\Theta}$ 进行迭代并找到更新的 $\Delta\boldsymbol{\Theta}$，以最小化目标函数：

$$\begin{aligned} E(\Delta\boldsymbol{\Theta}) &= \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{\Theta}+\Delta\boldsymbol{\Theta})-\boldsymbol{x}'_i\|^2 \\ &\approx \sum_i \|\boldsymbol{J}(\boldsymbol{x}_i;\boldsymbol{\Theta})\Delta\boldsymbol{\Theta}-\boldsymbol{r}_i\|^2 \\ &= \Delta\boldsymbol{\Theta}^{\mathrm{T}}[\sum_i \boldsymbol{J}^{\mathrm{T}}\boldsymbol{J}]\Delta\boldsymbol{\Theta}-2\Delta\boldsymbol{\Theta}^{\mathrm{T}}[\sum_i \boldsymbol{J}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{i}}]+\sum_i \|\boldsymbol{r}_i\|^2 \\ &= \Delta\boldsymbol{\Theta}^{\mathrm{T}}\boldsymbol{A}\Delta\boldsymbol{\Theta}-2\Delta\boldsymbol{\Theta}^{\mathrm{T}}\boldsymbol{b}+c \end{aligned} \tag{3.11}$$

其中，$\boldsymbol{J}$ 为 $\boldsymbol{f}$ 的雅可比矩阵，且

$$\boldsymbol{A}=\sum_i \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{x}_i)\boldsymbol{J}(\boldsymbol{x}_i) \quad \boldsymbol{b}=\sum_i \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{x}_i)\boldsymbol{r}_i \tag{3.12}$$

一旦计算出 $\boldsymbol{A}$ 和$\boldsymbol{b}$，即可用下式计算 $\Delta\boldsymbol{\Theta}$：

$$\boldsymbol{A}\Delta\boldsymbol{\Theta}=\boldsymbol{b} \tag{3.13}$$

并相应更新参数向量$\boldsymbol{\Theta}$ 为$\boldsymbol{\Theta}+\Delta\boldsymbol{\Theta}$。

可见，采用非线性最小二乘法的关键是计算变换的雅可比矩阵 $\boldsymbol{J}$。对单应性矩阵而言，$\boldsymbol{\Theta}=\boldsymbol{h}$，$\boldsymbol{f}$ 为式(3.8)的形式，其雅可比矩阵为

$$\boldsymbol{J}(\boldsymbol{x}_i)=\frac{\partial\boldsymbol{f}(\boldsymbol{x}_i)}{\partial\boldsymbol{h}}=\frac{1}{D_i}\begin{pmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_ix_i & -x'_iy_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_ix_i & -y'_iy_i & -y'_i \end{pmatrix} \tag{3.14}$$

其中 $D_i=h_{20}x_i+h_{21}y_i+h_{22}$。注意，式(3.14)中 $D_i$、$x'_i$、$y'_i$的计算都依赖 $\boldsymbol{h}$，可以基于 $\boldsymbol{h}$ 的当前值进行计算。

### 3.2.3　从单应性矩阵分解旋转和平移

对于场景中的一个平面物体而言，在获得其相对于图像的单应性矩阵 $\boldsymbol{H}$ 之后，还可以进一步分解得到该平面物体在相机坐标系中的旋转和平移参数。假设相机的内参矩阵 $\boldsymbol{K}$ 已知，则根据式(3.3)可得

$$(\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t}) \sim \boldsymbol{K}^{-1}\boldsymbol{H} \tag{3.15}$$

不妨设 $\boldsymbol{r}_1$、$\boldsymbol{r}_2$、$\boldsymbol{t}$ 分别为矩阵 $\boldsymbol{K}^{-1}\boldsymbol{H}$ 的 3 个列向量。又由于旋转矩阵 $\boldsymbol{R}$ 是正交矩阵，因此可以得到 $\boldsymbol{r}_3=\boldsymbol{r}_1\times\boldsymbol{r}_2$，这样便得到了 $\boldsymbol{R}$、$\boldsymbol{t}$ 的初始估计。

不过，由于对应点误差和计算误差的存在，按照上述方法直接计算得到的 $\boldsymbol{r}_1$、$\boldsymbol{r}_2$ 不一定是正交的单位向量，因此不能保证 $\boldsymbol{R}$ 是一个旋转矩阵。为此，首先对 $\boldsymbol{H}$ 乘以一个系数 $\lambda$：

$$(\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{t})=\lambda\boldsymbol{K}^{-1}\boldsymbol{H} \tag{3.16}$$

其中 $\lambda$ 的取值应使 $\boldsymbol{r}_1$、$\boldsymbol{r}_2$ 尽量接近单位向量。记 $\lambda_1=1/\|\boldsymbol{K}^{-1}\boldsymbol{h}_1\|$，$\lambda_2=1/\|\boldsymbol{K}^{-1}\boldsymbol{h}_2\|$，其中 $\boldsymbol{h}_1$、$\boldsymbol{h}_2$ 分别为 $\boldsymbol{H}$ 的第一个和第二个列向量，则可以取 $\lambda=0.5(\lambda_1+\lambda_2)$。注意这里再次利用了单应性变换的齐次性，对 $\boldsymbol{H}$ 而言，乘以 $\lambda$ 并不会改变其所表示的变换。

为了最终能够获得一个正交的旋转矩阵，就必须以上述方法所得结果为初值，求解一个离该结果最近的正交矩阵作为最终的旋转矩阵：

$$\min_{\boldsymbol{R}} \|\boldsymbol{R}-\boldsymbol{Q}\|_{\mathrm{F}}^{2} \quad \text{s.t.} \quad \boldsymbol{R}^{\mathrm{T}}\boldsymbol{R}=\boldsymbol{I} \tag{3.17}$$

其中 $\boldsymbol{Q}=(\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_1\times\boldsymbol{r}_2)$是根据式(3.15)计算得到的结果，$\|\cdot\|_{\mathrm{F}}$ 是矩阵的 Frobenius 范数，即矩阵各元素的平方和再开方。上述问题的最优解可以通过对矩阵 $\boldsymbol{Q}$ 的 SVD 得到，记 $\boldsymbol{Q}=\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{\mathrm{T}}$，其中 $\boldsymbol{S}$ 为对角阵，可以证明最优解为 $\boldsymbol{R}^{*}=\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}$。

## 3.3　平面标志的检测

在 3.2 节中，假定已知一组平面标志到图像上的对应点对，就可以求解出平面标志在相机坐标系中的 3D 位姿。为了求解单应性矩阵，最少需要 4 组点对。对于如图 3.1 所示的正方形或矩形平面标志而言，一般选择正方形或矩形的 4 个角点建立对应。4 个角点在平面标志上的坐标可以根据平面标志的尺寸提前设定，并在系统运行过程中保持不变。因此，主要问题是获取平面标志的 4 个角点在图像上的对应像素坐标，这也是对平面标志进行检测的主要目的。

根据平面标志设计特点的不同，对其进行检测的方法也会有所不同。对于如图 3.1 所示的正方形或矩形平面标志而言，其检测方法一般包含两个关键步骤：第一步是检测出图像中的候选四边形区域，并定位四边形的角点和边；第二步是根据每个候选四边形区域内部的图像内容，识别该四边形是否为平面标志，以及平面标志的类型(对包含多种标志的情况而言)。下面以平面标志检测库 ARUCO 中采用的检测算法为例，进一步说明上述两个步骤的具体实现过程。如图 3.3 所示为 ARUCO 中的平面标志检测算法工作的主要过程。

获得视频输入帧以后，首先将其转换成灰度图像，然后进行边缘检测或阈值化处理，获得如图 3.3(b)所示的边缘图像。边缘图像是一张二值图像，其中白色像素表示强边缘所在的位置。由于平面标志一般都被设计为高对比度的黑白图案，因此组成平面标志的区域边

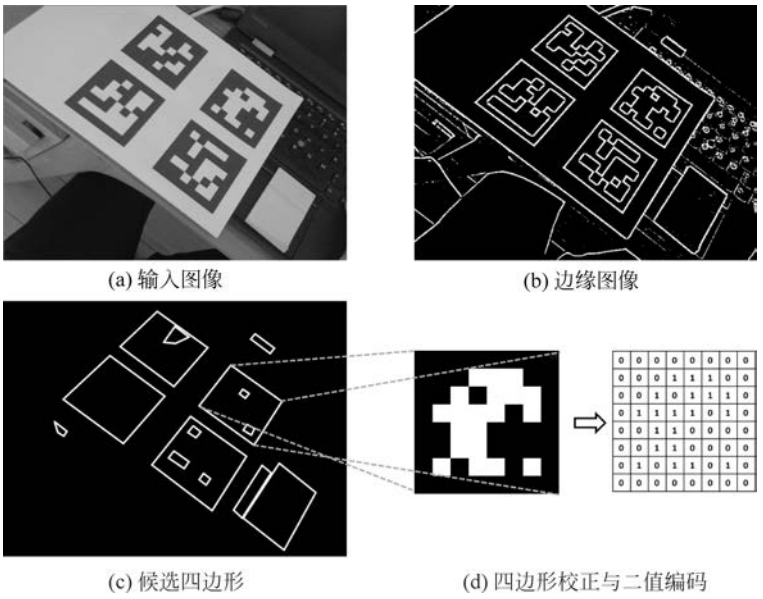



图 3.3 平面标志的检测过程

缘可以被较为稳定地检测到。相比于 Canny 边缘检测，阈值化处理一般速度更快，且运动模糊会更稳定，缺点是对边缘的定位精度相对较低。图中所示图像为用 OpenCV 中的局部自适应阈值函数 adaptiveThreshold 计算获得的结果。

获得边缘图像后，可以进一步提取候选的四边形，结果如图 3.3(c)所示。这可通过两个步骤完成。第一步是用轮廓跟踪算法找出边缘图中的闭合轮廓，并将轮廓边缘的像素连接为一个多边形。轮廓跟踪可以采用 Suzuki-Abe 算法，该算法可采用 OpenCV 中的 findContours 函数来实现。第二步，为了从闭合轮廓中找出四边形，可以对表示闭合轮廓的多边形进行简化，将近似为直线的轮廓点用一条线段表示。如果简化后的轮廓只包含四条边，则可以认为是一个候选的四边形区域。多边形简化可以采用 Douglas-Peucker 算法，该算法可用 OpenCV 中的 approxPoly 函数实现。

最后，基于图像内容，剔除候选四边形中的非平面标志区域，并识别每个平面标志的类别 ID。这主要通过图 3.3(d)的二值编码来完成。对每个候选四边形来说，都应先将其 4 个顶点与图 3.3(d)所示的平面标志的标准形状建立对应。注意，由于无法仅根据形状确定具体的对应关系，因此需要对所有的 4 种可能逐一进行尝试。对每一种可能而言，都应根据 4 组点对应，采用 3.2.2 节介绍的方法求得从图像到标准形状的单应性变换 $\boldsymbol{H}$，再根据 $\boldsymbol{H}$ 将候选四边形内部的图像内容校正到标准形状。如果某个候选四边形是一个平面标志，且点对应正确，则经过 $\boldsymbol{H}$ 变换后的结果将与相应平面标志的标准模板对齐。为了度量对齐程度，应将经 $\boldsymbol{H}$ 变换后的图像区域转换成图 3.3(d)所示的二值编码。在系统初始化时，对系统中所有可能的平面标志，采用同样的方式计算其图像的二值编码并建立索引表。为了确定一个候选四边形区域是否为平面标志，可以将其二值编码在索引表中进行检索，找出与之最相似的一个平面标志。如果相似度足够高，则认为该候选是一个平面标志，并赋予相应平面标志的类别 ID。

可见，平面标志的特殊形状和颜色极大地简化了对其进行检测的过程，只需用基本的图像处理技术即可获得较为稳定的检测结果。在视频中，物体的运动一般是连续的，在视频的相邻帧之间不会有太大的差异。利用这个特点，还可以基于上一帧的平面标志位姿参数，跟踪出当前帧的平面标志位姿参数。相比于检测，跟踪只是进行局部搜索，因此一般来说可以更快、更精确。不过，当物体或者相机运动较快的时候，跟踪也容易因视频的连续性假设不满足而失败。因此，实际系统中往往需要检测和跟踪相互配合才能获得较好的效果。

## 3.4　虚拟物体绘制

为了获得最终的绘制输出，首先需要将相机内参矩阵 $\boldsymbol{K}$ 和物体位姿参数 $\boldsymbol{R}$ 和 $\boldsymbol{t}$ 转换成图形，并绘制引擎的相应参数。这里以 OpenGL 为例，说明相应的转换方法。

在 OpenGL 中，相机内参矩阵 $\boldsymbol{K}$ 的作用等价于投影变换矩阵。回顾 2.3.1 节，如果不考虑成像面的倾斜角，则内参矩阵 $\boldsymbol{K}$ 可以写为如下形式：

$$\boldsymbol{K}=\begin{pmatrix}\alpha_x & 0 & x_0\\ 0 & \alpha_y & y_0\\ 0 & 0 & 1\end{pmatrix} \tag{3.18}$$

不过，上述形式的内参矩阵 $\boldsymbol{K}$ 与 OpenGL 要求的投影变换矩阵是不一致的，无法直接设置。在 OpenGL 中，投影变换矩阵 $\boldsymbol{P}$ 是一个 $4\times4$ 的矩阵，除了表示 3D 到 2D 的投影关系外，还包含对深度值的变换。从 $\boldsymbol{K}$ 到 $\boldsymbol{P}$ 的变换可以采用如下形式：

$$\boldsymbol{P}=\begin{pmatrix}\dfrac{2\alpha_x}{w} & 0 & 1-\dfrac{2x_0}{w} & 0\\ 0 & -\dfrac{2\alpha_y}{h} & 1-\dfrac{2y_0}{h} & 0\\ 0 & 0 & -\dfrac{Z_1+Z_0}{Z_1-Z_0} & -\dfrac{2Z_1Z_0}{Z_1-Z_0}\\ 0 & 0 & -1 & 0\end{pmatrix} \tag{3.19}$$

其中，$w$ 和 $h$ 分别为图像的宽和高，$Z_0$ 和 $Z_1$ 分别为 OpenGL 中近裁剪面和远裁剪面所对应的深度值(具体可参考 gluPerspective 函数)。

根据 $\boldsymbol{R}$ 和 $\boldsymbol{t}$ 计算 OpenGL 的模型-视图变换较为简单。但需要注意的是，由于在图像坐标系一般假设 $y$ 轴向下，$x$ 轴向右，导致引出的 $z$ 轴向里(远离视点)；而在 OpenGL 中，图像坐标系的 $y$ 轴向上，$z$ 轴向外，因此，在图像坐标系中计算 $\boldsymbol{R}$ 和 $\boldsymbol{t}$ 转换到 OpenGL 坐标系时，需要将 $y$ 坐标和 $z$ 坐标反向，相应的模型-视图变换为

$$\boldsymbol{T}=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\boldsymbol{R} & \boldsymbol{t}\\ 0 & 1\end{pmatrix} \tag{3.20}$$

采用上述公式获得投影矩阵 $\boldsymbol{P}$ 和模型-视图变换矩阵 $\boldsymbol{T}$ 之后，就可以采用 glLoadMatrix 函数设置变换矩阵到绘制管线，再进一步完成绘制。为了将虚拟物体直接绘制到图像中，可以将背景图像作为纹理贴图先进行绘制。

最后，总结一下本章所述的简易增强现实系统。其基本流程是，对每一帧输入图像，首先用 3.3 节的方法检测平面标志，定位出平面标志的 4 个角点；其次用 3.2 节的方法，通过点对估计平面标志的 3D 位姿；最后用 3.4 节的方法绘制虚拟物体。

上述过程主要解决的是虚拟物体在现实空间的注册问题，其中使用了特殊设计的平面标志作为辅助，并将虚拟物体的位姿与平面标志的局部坐标系相绑定。平面标志的优点是其特征点非常明显，易于检测。本书接下来的几章将讨论更一般的空间注册方法，其 3D 特征点坐标的获取各具特色，第 4 章介绍由各类传感器通过实时测量获取 3D 坐标的方法，第 5 章通过对自然图像上特征点进行检测和匹配来获取 3D 点坐标，第 6 章则采用视觉方法通过对视频图像序列上特征点的匹配来重构特征点的 3D 坐标。

## 扩展阅读

增强现实系统开发涉及图像视频处理、计算机视觉、计算机图形学等方面的专业知识，同时为了满足实时性，对代码的计算效率有较高的要求。实际项目中往往都基于已有的增强现实 SDK 实现底层核心功能。ARToolKit 是一个较为早期的增强现实 SDK，最早发布于 1999 年，具有开源、免费的优势，可以支持 Windows、Linux 和 OS X，但不支持移动平台。近年来，随着手机、平板电脑等移动设备的普及，增强现实的主流应用逐渐转移到了移动计算平台。目前常用的增强现实 SDK 有高通公司的 Vuforia、谷歌公司的 ARCore、苹果公司的 ARKit，以及视辰信息科技公司的 EasyAR 等。这些工具包都支持增强现实的基本核心功能，如相机标定、平面标志物的识别跟踪、基于 SLAM 技术的相机跟踪与场景 3D 重建，以及图形绘制等。关于这些工具包的更多信息都可以通过搜索引擎找到，感兴趣的读者可以自行查阅。

## 习题

1. 已知 $\boldsymbol{X}=(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z})^{\mathrm{T}}$ 是物体 3D 模型上的一个点，物体在相机坐标系中的旋转和平移参数分别为 $\boldsymbol{R}$、$\boldsymbol{t}$，相机内参矩阵为 $\boldsymbol{K}$。求 $\boldsymbol{X}$ 在图像上的投影点坐标 $\boldsymbol{x}=(x,y)^{\mathrm{T}}$。

2. 如果单应性矩阵 $\boldsymbol{H}$ 表示某 3D 平面到图像的变换，则 $\boldsymbol{H}$ 主要包含哪些信息？

3. $\pi$ 是一个 3D 平面，$I_1$ 和 $I_2$ 分别是平面 $\pi$ 在不同视角下观测到的图像，证明：平面 $\pi$ 在 $I_1$ 和 $I_2$ 中的投影像素坐标可以通过一个单应性矩阵进行精确对应，即存在单应性矩阵 $\boldsymbol{H}$，使得 $\forall \boldsymbol{X}\in\pi$，有 $\tilde{\boldsymbol{x}}_1\sim\boldsymbol{H}\tilde{\boldsymbol{x}}_2$，其中 $\tilde{\boldsymbol{x}}_1$ 和 $\tilde{\boldsymbol{x}}_2$ 分别是 $\boldsymbol{X}$ 在 $I_1$ 和 $I_2$ 中投影像素的齐次坐标。

4. 推导单应性矩阵的雅可比矩阵 $\boldsymbol{J}$，即式(3.14)。

5. 理解从 $\boldsymbol{K}$ 到 OpenGL 投影变换矩阵 $\boldsymbol{P}$ 的变换公式，即式(3.19)。

6. 根据本章所述基本原理，实现一个如图 3.1 所示的简易增强现实系统。