第3章

视 频 处 理

3.1 引言

视频是图像和音频的组合，因此一般而言，只要对图像和音频有效的理论和应用，通常也对视频有效。视频由一组称为**帧**的静止图像组成，这些图像以称为**帧率**的特定速度一个接一个地显示给用户，以每秒帧数为单位，缩写为 fps。如果以足够快的速度显示，人眼无法将单幅图像区分为单独的实体，而是将连续图像合并在一起，从而产生活动图像的错觉，这种现象称为**视觉暂留**(PoV)。已经观察到帧率应该在 25～30fps，以便让人眼感知没有间隙或抖动的平滑运动。添加音频并与图像的活动同步可以创建完整的视频序列。因此，一个视频文件是由多个图像帧和一个或多个音轨组成。同时处理如此多信息的一个缺点是文件大小增加，需要大量处理资源来处置它们。例如，一个一分钟的视频文件由 30 帧组成，每帧大小为 640×480 像素，并使用 24 位彩色信息，则占用的空间超过 1582MB。以 44 100Hz 采样的音频每分钟为文件增加 10MB。此外，播放视频文件需要大约 30MB/s 的带宽。因此，压缩方案对于视频处理如此大的开销非常重要。

为了创建数字视频，我们首先需要将视觉和音频信息以电信号的形式记录在磁带或磁盘上。用于指定这种表示形式的术语是**运动视频**，以将其与电影院中使用的另一种称为**运动图片**的表示形式区分开，后者在电影院中使用光化学过程将视频帧记录到赛璐珞胶片上。电子信号形式的运动视频由**模拟摄像机**生成并存储在磁带(如录像带)中，然后使用录像带播放器(VCP)进行播放。电视传输也是运动视频显示的流行示例。早期的模拟摄像机使用称为**阴极射线管**(CRT)的真空管来生成这些信号，然后可以将这些信号馈送到监视器以显示视频，而音频则使用麦克风单独录制并馈送到扬声器以生成声音。单色或灰度视频需要来自摄像机的单个强度信号作为视觉信息以及一两个音频信号，具体取决于播放的声音是单声道还是立体声。为了在 CRT 监视器屏幕上显示图像，来自阴极的电子束被激活并聚焦在涂有荧光粉的屏幕上发光。**磷光体**是一种化学物质，当它与电子等带电粒子接触时会发出光芒。为了在屏幕上生成图像，电子束从屏幕的左上角开始，从左到右依次扫描第一行荧光点。在每条水平线的末尾，光束对角移动到下一行的开头并开始跟踪操作。在右下

角，光束对角移动到左上角的起点，并再次重复该操作。这个过程称为**光栅扫描**，通常每秒需完成大约 60 次以获得屏幕上稳定的画面，这称为显示器的刷新率，屏幕上产生的每幅图像称为一**帧**。支持 60 帧/秒的监视器会产生不闪烁的图像，称为**逐行扫描监视器**。另一种技术，主要用于刷新率较低的监视器，称为**隔行扫描**，相应的监视器称为**隔行扫描监视器**。在该情况下，一帧被分成两半，每一半称为一个**场**。第一个由奇数行组成的场称为奇数场，第二个由偶数行组成的场称为偶数场。每个场只包含一半的行数，每秒扫描 60 次，从而将有效刷新率降低到 30 帧/秒。由于 PoV，这种处理可使每个场中的行平滑混合，并有助于在低刷新率下生成不闪烁的图像。字母“p”和“i”用于区分逐行扫描和**隔行扫描**监视器，例如 720p 和 1080i，数字表示监视器中水平行的总数。

新一代摄像机用称为**电荷耦合器件**(CCD)的电子光电传感器取代 CRT，CCD 产生的电信号大致与落在 CRT 上的光强度成正比。将来自 CCD 阵列的信号按顺序收集并发送到监视器进行显示。现代显示器使用**液晶显示器**(LCD)器件代替 CRT 和电子束。LCD 器件是一种小的透明块，填充有由长棒状分子组成的液态有机化学物质，具有操纵光线通过该物质的方向的特性。LCD 器件和一对偏振滤光片允许光线从背光源(如 LED)流向前方的观察者，从而产生发光像素的感觉。流过 LCD 器件的电流可改变分子的方向，阻止光线到达观察者。打开和关闭特定点有助于在屏幕上创建图像。对于彩色摄像机，三个单独的 **RGB 信号**对应于红色、绿色和蓝色这三种基色，用于在屏幕上创建合成彩色，这些信号通过三条单独的电缆馈送到彩色监视器，这一方案被称为**分量视频**。在显示器内部，这些信号用于激活彩色生成系统，如 CRT 电子枪或用于彩色再现的 LCD 器件。

用于彩色再现的 RGB 信号在短距离传输时效果很好，典型情况是若干米。然而，当这些信号需要像**电视传输**那样跨越几千米的远距离传输时，工程师遇到了一些问题。首先，三根独立的铜缆运行数千米，使系统成本高昂。其次，即使忽略成本，由于衰减系数不同，沿三根电缆传输的三个独立信号也不会完全在同一时刻到达接收端，经常导致图像不同步和扭曲。再次，当彩色电视刚开始在几个国家传输时，早期的黑白(B/W)电视的单色系统也还在继续使用，因此工程师不得不想出一个系统，以便相同的传输信号可以满足同时连接到黑白电视机和彩色电视机，这是使用现有的 RGB 信号格式无法实现的。为了解决所有这些问题，开发了一种称为**复合视频**的新信号格式，它使用一种形式不同的称为 YC 的信号，这里，Y 表示**亮度**或强度信号，C 表示**色度**或彩色信号。这种格式相对于 RGB 格式的优点为 Y 和 C 都可以通过单个电缆或通道传输，因此可在同一时刻到达接收端。YC 信号是通过将电视频道的典型 6MHz 带宽分成两部分来实现，0～4MHz 分配给 Y，1.5MHz 分配给 C，其余 0.5MHz 用于音频。这种不均匀分布背后的原因是人眼对 Y 信息更敏感，而对 C 信息不太敏感。YC 信号格式的另一个优点是，可以只将 Y 信号送给黑白电视机，而将 Y 和 C 信号都送给彩色电视机。这意味着只需要一个单一的传输系统，并且在接收端只需要一个滤波器就可以去除彩色信号以进行单色观看。由于使用单根传输电缆，还可以降低系统成本。

可以使用 RGB 信号的线性组合来计算代表灰度强度的 Y 信号。在尝试了多种组合并考虑人眼对色谱的绿色部分更敏感后，最终决定 Y 由 60%的 G、30%的 R 和 10%的 B 组成，得到关系式为

$$Y = 0.2989R + 0.5870G + 0.1140B$$

彩色信息使用圆形刻度而不是线性刻度表示，需要两个组件来将特定彩色值标识为平

面上的一个点，即色轮。两个彩色子成分分别称为 C_b 和 C_r，定义如下：

$$C_b = B - Y$$

$$C_r = R - Y$$

因此，需要单一电缆或通道进行传输的复合视频格式特指使用 $\mathbf{YC_bC_r}$ 信号格式。当今用于传输视频信号的大多数模拟视频设备使用复合视频电缆和连接器作为接口，例如在 VCP 和 TV 之间。

RGB 到 YC 信号格式的转换还有另一个重要优势：通过减少彩色信息可以使用更少的带宽。研究表明，人眼对亮度信息比对色度信息更敏感。这一发现被用来减少视频传输过程中的彩色信息，该过程被称为**色度子采样**。彩色信息的减少由一组三个数字表示，表示为 $A:B:C$。数字表示屏幕上窗口内的亮度和色度信息量，通常为 4 像素宽和 2 像素高。常见的值包括 4∶2∶2，这意味着在一个滑动的 4×2 窗口中，第一行有 4 个包含 Y 信息的像素，有 2 个包含 C 信息的像素，第二行有 2 个包含 C 信息的像素。本质上，这意味着虽然所有像素都包含亮度信息，但彩色信息沿水平方向减少了一半。其他常用的值为 4∶1∶1 和 4∶2∶0。第一组表示水平方向的四分之一彩色信息，而第二组表示沿水平和垂直方向的半色信息。显然，4∶4∶4 的值表示彩色信息没有减少。

与图像和音频一样，称为 CODEC(编/解码器)的压缩方案可用于减小数字视频文件的大小。由于视频文件比较大，无损算法用得不多。常采用有损压缩算法从图像和音频组件中删除信息以减小视频文件的大小。保存视频的**文件格式**取决于所使用的压缩方案。Windows 本机音频文件格式是 AVI，通常未压缩。有损压缩算法与用于移动平台的 MPEG(MPEG-1)、Window Media Video(WMV)、MPEG-4(MP4)、Apple Quicktime Movie(MOV)和第三代合作伙伴计划(3GPP)等文件格式相关联。

3.2　工具箱和函数

MATLAB 中的视频处理函数可以分为两类：基本 MATLAB(BM)函数和计算机视觉系统工具箱(CVST)函数。BM 函数是一组用于执行初步数学矩阵运算和图形绘制操作的基本工具。CVST 函数提供用于设计和模拟计算机视觉和视频处理系统的算法、函数和应用程序，包括一组更高级的工具，用于专门处理任务，如团点分析、目标检测和运动跟踪。有些函数，例如导入/导出和基本播放，对于 BM 集和 CVST 集都是通用的。为了解决某些特定任务，可能同时需要来自 BM 集和 CVST 集的函数。这些函数的来源在本书中用于说明示例时会相应地被提及。本章中，对用于视频处理任务的 MATLAB 特性，借助特定示例的解决方案进行了说明。本书使用 MATLAB 2018b 版编写，但是，大部分功能都可以在 2015 及其后的版本中使用，几乎没有更改。MATLAB 支持的视频文件格式包括 MPEG-1(.mpg)、Windows Media Video(.wmv)、包括 H.264 编码的 MPEG-4 视频(.mp4、.m4v)、Apple QuickTime Movie(.mov)、3GPP 和 Ogg Theora(.ogg)。示例中使用的大部分视频文件都包含在 MATLAB 软件包中，不需要用户提供。内置媒体文件包含在以下文件夹中：(matlab-root)\toolbox\vision\visiondata。

3.2.1　基本 MATLAB(BM)函数

本章中使用的 BM 函数分为五个不同的类别：语言基础、图形、数据导入和分析、编程脚

本和函数、高级软件开发。下面提供了这些函数的列表和它们的层次结构以及对每个函数的一行描述。BM 集由数千个函数组成，考虑本书的范围和内容，本章使用了其中的一个子集。

1. 语言基础

clc：clear command window，清除命令窗口

2. 图形

(1) **figure**：create a figure window，创建图形窗口

(2) **frame2im**：return image data associated with movie frame，返回与电影帧关联的图像数据

(3) **getFrame**：capture axes or figure as movie frame，捕获轴或图形作为电影帧

(4) **im2frame**：convert image to movie frame，将图像转换为电影帧

(5) **plot**：two-dimensional(2-D) line plot，二维(2-D)线图

(6) **subplot**：multiple plots in a single figure，单个图片中的多个绘图

(7) **title**：plot title，绘制标题

(8) **xlabel**，**ylabel**：label x-axis，y-axis，x 轴标签、y 轴标签

3. 数据导入和分析

(1) **clear**：remove items from workspace memory，从工作区内存中删除项目

(2) **hasFrame**：determine if frame is available to read，确定帧是否可供读取

(3) **mmfileinfo**：information about multimedia file，多媒体文件信息

(4) **readFrame**：read video frame from video file，从视频文件中读取视频帧

(5) **VideoReader**：read video files，读取视频文件

(6) **VideoWriter**：write video files，写入视频文件

4. 编程脚本和函数

(1) **for ... end**：repeat statements specified number of times，重复语句指定次数

(2) **load**：load variables from file into workspace，将文件中的变量加载到工作区

(3) **pause**：stop MATLAB execution temporarily，暂时停止 MATLAB 执行

(4) **while ... end**：execute statements while condition is true，条件为真时执行语句

5. 高级软件开发

(1) **release**：release resources，释放资源

(2) **step**：run system object algorithm，运行系统对象算法

3.2.2 计算机视觉系统工具箱(CVST)函数

CVST 提供用于设计和模拟计算机视觉和视频处理系统的算法、函数和应用程序。CVST 包括用于修改数字视频信号的视频处理算法库。它们分为三类：输入、输出和图形；目标检测和识别；目标跟踪和运动估计。下面列出了本章讨论的 CVST 中的一些最常用函数：

1. 输入、输出和图形

(1) **insertObjectAnnotation**：annotate image or video，注释图像或视频

(2) **insertText**：insert text in image or video，在图像或视频中插入文本

(3) **vision. VideoFileReader**：read video frames from video file，从视频文件中读取视频帧

（4）**vision. VideoPlayer**：play video or display image，播放视频或显示图像

（5）**vision. DeployableVideoPlayer**：display video，显示视频

（6）**vision. VideoFileWriter**：write video frames to video file，将视频帧写入视频文件

2. 目标检测和识别

（1）**vision. BlobAnalysis**：properties of connected regions，连通区域的性质

（2）**vision. ForegroundDetector**：foreground detection using Gaussian mixture models (GMM)，使用高斯混合模型(GMM)进行前景检测

（3）**vision. PeopleDetector**：detect upright people using histogram of oriented gradient (HOG) features，使用朝向梯度直方图(HOG)特征检测直立的人

（4）**vision. CascadeObjectDetector**：detect objects using the viola-Jones algorithm，使用 Viola-Jones 算法检测物体

（5）**ocr**：recognize text using optical character recognition(OCR)，使用光学字符识别(OCR)识别文本

3. 目标跟踪和运动估计

（1）**configureKalmanFilter**：create Kalman filter for object tracking，创建用于目标跟踪的卡尔曼滤波器

（2）**opticalFlowFarneback**：estimate optical flow using Farneback method，使用 Farneback 方法估计光流

（3）**opticalFlowHS**：estimate optical flow using Horn-Schunck method，使用 Horn-Schunck 方法估计光流

（4）**opticalFlowLK**：estimate optical flow using Lucas-Kanade method，使用 Lucas-Kanade 方法估计光流

（5）**opticalFlowLKDoG**：estimate optical flow using Lucas-Kanade derivative of Gaussian method，使用高斯方法的 Lucas-Kanade 导数估计光流

（6）**vision. BlockMatcher**：estimate motion between images or video frames，估计图像或视频帧之间的运动

（7）**vision. HistogramBasedTracker**：histogram-based object tracking，基于直方图的目标跟踪

（8）**vision. PointTracker**：track points in video using Kanade-Lucas-Tomasi (KLT) algorithm，使用 Kanade-Lucas-Tomasi (KLT)算法跟踪视频中的点

（9）**vision. KalmanFilter**：Kalman filter for object tracking，用于目标跟踪的卡尔曼滤波器

3.3　视频输入输出和播放

播放视频文件最基本的方法是使用指定文件名的 IPT 函数 **implay**。BM 函数 **mmfileinfo** 用于提取有关文件的信息，例如音频和视频结构，见例 3.1 中为方法 1。BM 函数 **VideoReader** 用于读取视频文件，从而创建 VideoReader 对象。对象的属性可用于检索视频信息，如持续时间、帧高度和宽度、文件路径、帧速率、帧数、文件格式等。属性

getFileFormats 返回支持的视频文件格式。读取方法可用于读取存储在包含帧高度、帧宽度、彩色通道索引和帧索引的 4-D 数组中的视频数据。然后可以使用 IPT 函数 implay 来播放视频数据。为了播放帧的子集，可以指定开始帧和结束帧。结束帧为 Inf 表示播放视频直到结束，见例 3.1 中的方法 2(见图 3.1)。

例 3.1 编写一个程序，读取和播放视频文件。

```
clear; clc;
fv = 'rhinos.avi';
fa = 'RockDrums-44p1 - stereo - 11secs.mp3';
% method - 1
implay(fv);
ia = mmfileinfo(fa); ia.Audio
iv = mmfileinfo(fv); iv.Video
% method - 2
vr = VideoReader(fv);              % video reader object
d = vr.Duration
h = vr.Height
w = vr.Width
p = vr.Path
bp = vr.BitsPerPixel
fr = vr.FrameRate
nf = vr.NumberOfFrames
fo = vr.VideoFormat
ff = vr.getFileFormats
v = read(vr);                  % 4 - D array containing video frames
implay(v)
sf = 11;                       % start frame
ef = 50;                       % end frame
cv1 = read(vr,[sf, ef]);
cv2 = read(vr,[50, Inf]);
implay(cv1)                    % Total frame numbers indicated in status bar
implay(cv2)                    % Total frame numbers indicated in status bar
```

图 3.1 例 3.1 的输出

另外，也可以使用 readFrame 方法单独读取每帧并使用循环播放它们。返回的每帧(vf)是一个关于彩色图像的 3-D 数组，根据要显示的总帧数重复该操作。可以从视频对象的属性 NumberOfFrames 中提取总帧数。视频帧在图形窗口中显示为一系列图像，每幅图像之间的停顿是帧率的倒数，即连续帧之间的时间间隔。计数器可用于在图像窗口的标题字段中显示相应的帧编号。当视频需要从指定的偏移时间而不是从开始播放时，将使用此句法。创建视频对象时，可以使用选项 CurrentTime 指定播放的开始时间。在这种情况下，由于不计算就无法知道要播放的总帧数，因此显示帧的循环需要使用 while 结构而不是 for 结构，并使用 hasFrame 方法来确定帧是否可用于播放。例 3.2 显示了视频的两个实例：一个用于播放所有帧；另一个用于播放距开头 2.5s 偏移量的视频。实际播放的帧数显示在每幅图像的顶部。在第一种情况下，所有 114 帧都被播放；而在第二种情况下，播放的帧数可以通过从总持续时间中减去偏移时间并将结果乘以帧速率来计算，并四舍五入到下一个整数，即在这种情况下变为(7.6－2.5)×15≈77(见图 3.2)。

例 3.2 编写一个程序，使用起始时间偏移读取和播放视频帧的子集。

```
clear; clc;
fn = 'rhinos.avi';
vr = VideoReader(fn);      % video reader object
nf = vr.NumberOfFrames;    % number of frames
fr = vr.FrameRate;         % frame rate
du = vr.Duration;          % video duration
of = 2.5;                  % time offset in seconds from beginning
subplot(121)
vr = VideoReader(fn);
fi = 0;
for i = 1:nf
    vf = readFrame(vr);    % video frame
    fi = fi + 1;           % frame index number
    image(vf); title(fi); axis square;
    pause(1/fr);
end
subplot(122)
vr = VideoReader(fn, 'CurrentTime', of);
fi = 0;
while hasFrame(vr)
    vf = readFrame(vr);
    fi = fi + 1;
    image(vf); title(fi); axis square;
    pause(1/fr);
end
nof = (du - of) * fr;      % actual number of frames to be played
round(nof)
```

read 方法将视频存储在包含帧高度、帧宽度、彩色索引和帧索引的 4-D 数组中。可以操纵帧索引并选择总帧的子集进行显示和回放。例 3.3 显示了视频的每个 10 的倍数的帧并播放帧的子集。为了显示帧，创建了一个 3×3 的网格，其中，显示了第 10、20、30、40、50、

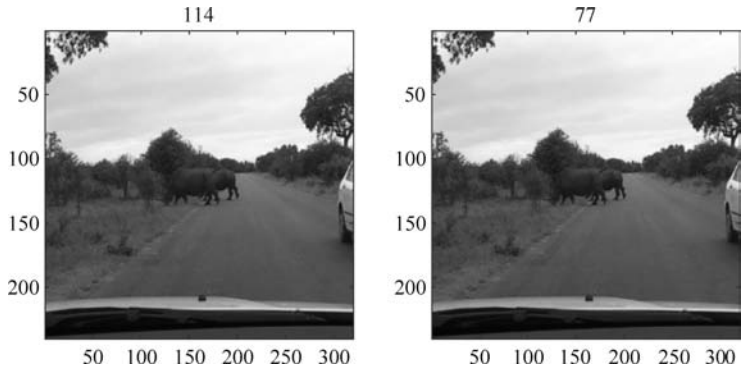

图 3.2　例 3.2 的输出

60、70、80、90 帧，这些帧被收集到一个新的 4-D 数组中并播放（见图 3.3）。

例 3.3　编写一个程序，显示每个 10 的倍数的帧并播放这些视频帧。

```
clear; clc;
vr = VideoReader('rhinos.avi');
v = read(vr);              % original video array
figure
f = 0;                     % frame index number
for r = 1:3                % row index of display grid
    for c = 1:3            % column index of display grid
        f = f + 10;
        subplot(3, 3, c + (r-1)*3), imshow(v(:,:,:,f)); title(f);
        w(:,:,:, c + (r-1)*3) = v(:,:,:,f);
    end
end
implay(w)                  % new video array
```

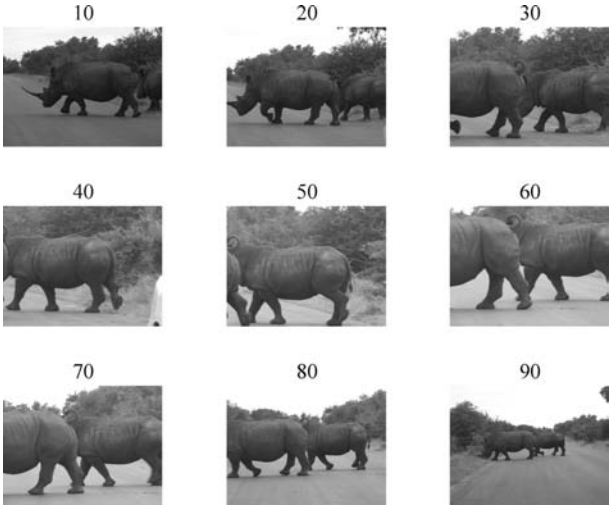

图 3.3　例 3.3 的输出

IPT 函数 **implay** 也可用于播放不构成视频一部分的单个图像序列，而 IPT 函数 **immovie** 可用于根据帧序列创建电影。例 3.4 在内存中加载了一系列图像并播放它们。对

于索引图像，要指定相应的彩色映射(见图 3.4)。

例 3.4　编写一个程序，播放一系列单幅图像并将一系列图像创建成电影。

```
clear; clc;
load mristack;
implay(mristack);
load mri;
mov = immovie(D,map);
implay(mov)
```

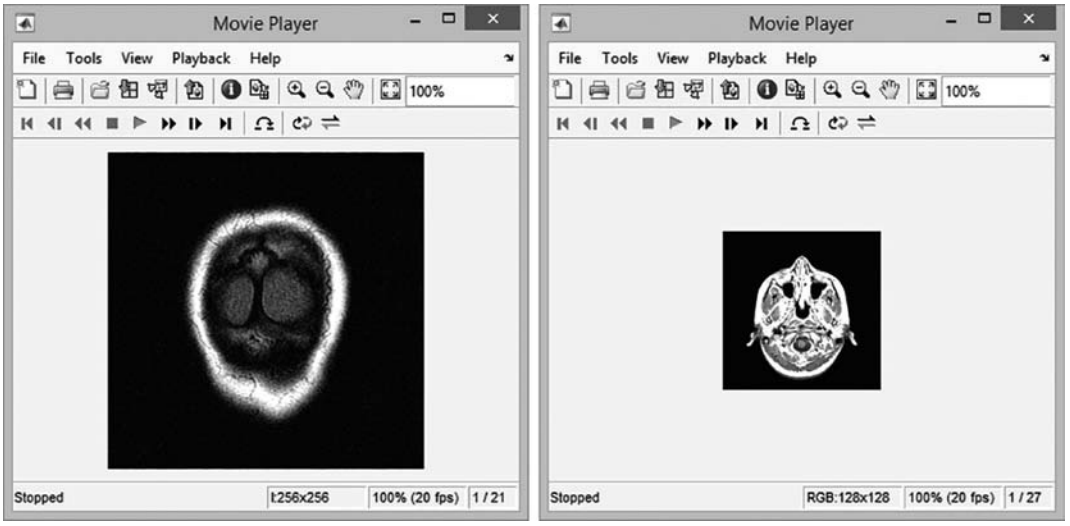

图 3.4　例 3.4 的输出

BM 函数 **getframe** 用于将当前图像捕获为视频帧。BM 函数 **movie** 用于将帧集合作为电影播放。例 3.5 根据 BM 函数 **peaks** 和 **surf** 创建了 3-D 表面，然后使用 BM 函数 **view** 通过将方位角和仰角从 1°更改为 360°来更改表面视图。这些视图中的每幅都被捕获为一个视频帧，然后按顺序播放(见图 3.5)。

例 3.5　编写一个程序，将一系列图像作为视频播放。

```
clear; clc;
for i = 1:360
j = 1;
surf(peaks); view(i,i); f(j) = getframe;
j = j+1;
end
movie(f)
```

read 方法会自动创建一个 4-D 结构来存储整个视频，而 readFrame 方法一次只读取一帧。要存储整个视频，可以使用 BM 函数 **struct** 通过指定帧尺寸、数据类型和彩色查找表(如果需要)显式创建结构。字段 cdata 包含 readFrame 返回的实际视频数据。例 3.6 说明了结构的创建，并显示了 BM 函数 **movie** 可用于通过指定帧速率和可选的重复值来播放存储在结构中的视频。BM 函数 **set** 用于根据屏幕坐标定位播放窗口(见图 3.6)。

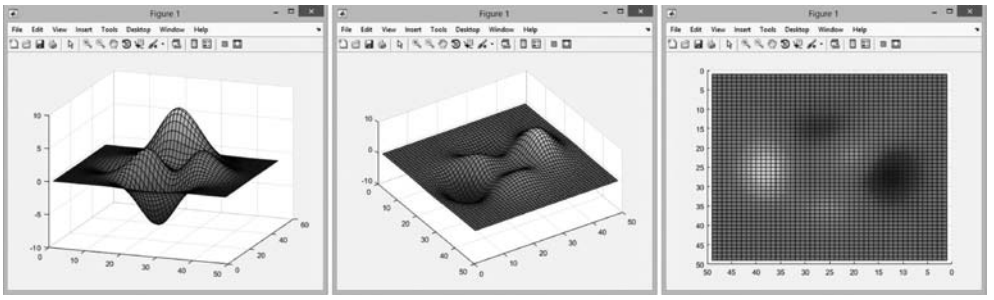

图 3.5　例 3.5 的输出

例 3.6　编写一个程序，创建电影结构并重复播放视频两次。

```
clear; clc;
vr = VideoReader('xylophone.mp4');
w = vr.Width;                          % frame width
h = vr.Height;                         % frame height
r = 2;                                 % repeat playback
m = struct('cdata',zeros(h,w,3,'uint8'),'colormap',[]);
 % read each frame and store in structure
k = 1;
while hasFrame(vr)
    m(k).cdata = readFrame(vr);
    k = k+1;
end
hf = figure;                           % figure handle
x = 150; y = 200;                      % position of playback window
set(hf,'position', [x y w h]);
movie(hf, m, r, vr.FrameRate);         % playback r times
```

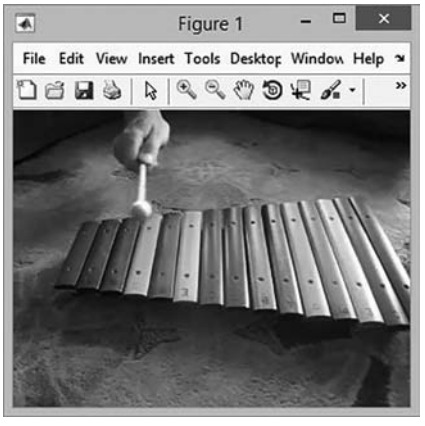

图 3.6　例 3.6 的输出

BM 函数 **VideoWriter** 可用于写入视频文件并创建 VideoWriter 对象。对象的属性可用于指定文件名、文件格式、压缩率、持续时间等。open 方法可以用来打开一个文件写入视频数据，close 方法可以用来写入视频数据后关闭文件，writeVideo 方法是将实际的视频数据写入文件。例 3.7 从随机数绘图创建视频帧并将这些帧写入视频文件。BM 函数