

学习 > Linux

概览

一、系统架构

二、JPEG 概述

三、JPEG 原理详细分析

四、JPEG 文件存储格式

五、JPEG 压缩过程的优化

六、JPEG 在本嵌入式 Linux 应用中遇到的问题

七、总结

一、系统架构

余涛

2008 年 7 月 04 日发布

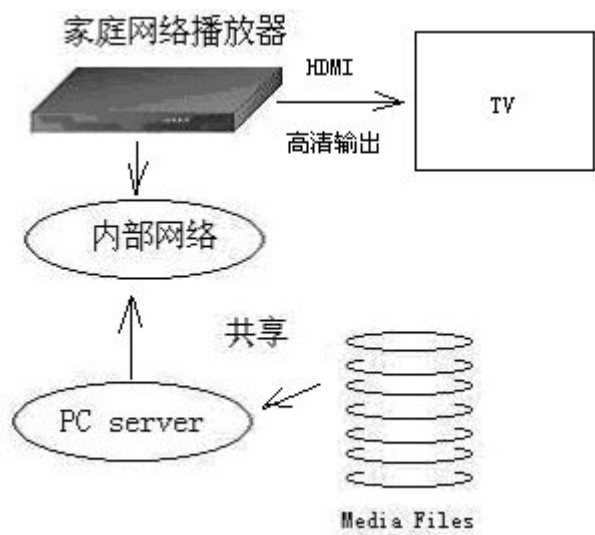
六、JPEG 在本嵌入式 Linux 应用中遇到的问题

七、总结

一、系统架构

相关主题

本文将以一个实际的产品为例，来说明 JPEG 在其中的应用。



本系统为一个嵌入式 Linux 网络播放器，主... 能为播放家庭网络中的多媒体文件

可以给用户提供更方便快捷的媒体文件的播放方式，并能充分利用家庭音响系统的巨大大改善了媒体文件的播放体验。

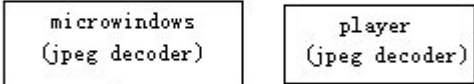
内容	各种图片的全屏播放
	图片的缩略图预览
概览	各种音频文件的播放
	音频专辑的封面图片显示
一、系统架构	各种视频文件的播放
	视频快照图片的显示
二、JPEG 概述	

三、JPEG 原理详细分析:

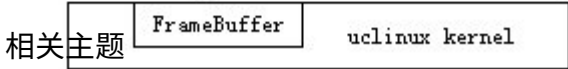
四、JPEG 文件存储格式

五、JPEG 压缩过程的优化

六、JPEG 在本嵌入式 Linux 应用中遇到的问题



七、总结



评论

本系统是基于嵌入式 Linux 的一个应用，使用的是 ucLinux 2.4.22，并使用了 microv kernel 的 FrameBuffer 作为显示输出。

此系统在两个方面使用到了 JPEG 库：

- 1、UI 的显示，即各种人机交互界面，考
- 户体验，所以大量使用了贴图来美化
- 2、JPEG 图片文件的全屏播放，包括用F
- 各种照片等

## 二、JPEG 概述

JPEG 是 Joint Photographic Experts Grc , 缩写，即 ISO 和 IEC 联合图像专家组

组开发的算法就被称为 JPEG 算法，并且已经成为了大家通用的标准，即 JPEG 标准。分是人的视觉不容易察觉到的部分，它充分利用了人眼对计算机色彩中的高频信息部

IBM Developer

学习    开发    社区

合作

内容

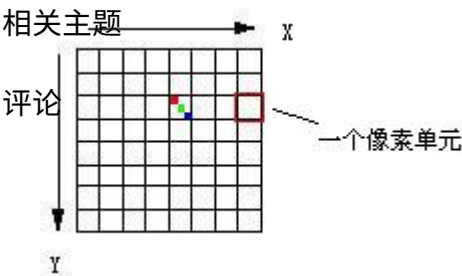
人眼对构成图像的不同频率成分具有不同的敏感度，这个是由人眼的视觉生理特性所细胞1.8亿个，含有对色彩敏感的椎状细胞0.08亿个，由于柱状细胞的数量远大于椎状对色彩的敏感程度。

- 概览
- 总体来说，一个原始图像信息，要对其进行 JPEG 编码，过程分两大步：
- 一、系统架构
    - 1、去除视觉上的多余信息，即空间冗余度
  - 二、JPEG 概述
    - 2、去除数据本身的多余信息，即结构（静态）冗余度

四、JPEG文件存储格式

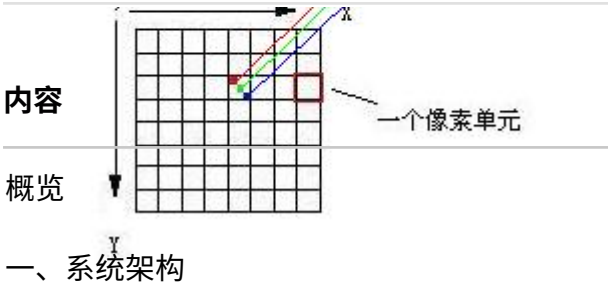
1、去除视觉上的多余信息

- 五、JPEG 压缩过程的优化
- 六、JPEG 在本嵌入式 Linux 应用中遇到的问题
  - 当你拿到一个原始未经处理的图像，是由各种色彩组成的，即在一个平面上，有各种很多点组成的。实际上，每个点的色彩，也即计算机能表示的每个像素点的色彩，能示总结即这三种颜色的一定比例的混合就能得到一个实际的色彩值。



所以，实际上，这个平面的图像，可以理各个分支 R/G/B 的混合时所占的具体数值的 R/G/B 三个值会比较接近。

了水平 X 和垂直 Y 以外，还有一个色每个像素的 RGB 的混合值可能都有



- 二、由于这个原始图像是由很多个独立的像素点组成的，也就是说它们都是分散的，离散示水平有640个像素点，垂直有480个像素点。
- 三、JPEG 原理详细分析
- 四、从上面的内容，我们可以知道两个相邻的点，会有很多的色彩是很接近的，那么如何不需要的数据，也即达到了压缩的效果。
- 五、JPEG 压缩过程的优化
- 这个就要涉及到图像信号的频谱特性了。
- 六、JPEG 在本嵌入式 Linux 应用中遇到的问题
- 图像信号的频谱线一般在0-6MHz范围内，而且一幅图像内，包含了各种频率的分量。
- 七、像区域比例很低的图像边缘的信号中才含有高频的谱线。这个是对 JPEG 图像压缩的

相关主题

因此具体的方法就是，在对图像做数字处理时，可根据频谱因素分配比特数：对包含对包含信息量低的高频谱区域分配较少的比特数，而图像质量并没有可察觉的损伤，

将原始图像这个色彩空间域，转换为频谱 Transform）变换。	么转呢，这个就用到了数学上的离散
DCT 是可逆的、离散的正交变换。变换过变换过程得到一个 DCT 变换系数，而对过用了。	虽然并不产生压缩作用，但是变换后[可以再进行更进一步的处理，即所谓
总体说来，这第一步，对图像进行编码，化（Quantization），这个过程就是依据据量，这个是结合数学方法与经验值而做	余的信息，要用到 DCT 变换中的正位直，来处理人眼视觉系统所不敏感的i，

## 2、去除数据本身的多余信息

IBM Developer    学习    开发    社区

合作

内容总体来说，上面的两步即：

概要如果处理的是彩色图像，JPEG 算法首先将 RGB 分量转化成亮度分量和色差分量，同半）。然后，用 DCT 来进行块变换编码，舍弃高频的系数，并对余下的系数进行量化一、系统架构程编码和 Huffman 编码来完成压缩任务。

二、JPEG 概述

## 三、JPEG 原理详细分析

### 四、JPEG文件存储格式

下面将更加详细地介绍这两步中的各个细节。

五、JPEG 压缩过程的优化

JPEG 编码中主要涉及到的内容主要包括：

六、JPEG 在本嵌入式 Linux 应用中遇到的问题

1. Color Model Conversion (色彩模型)

七、总结

2. DCT (Discrete Cosine Transform 离散余弦变换)

相关主题

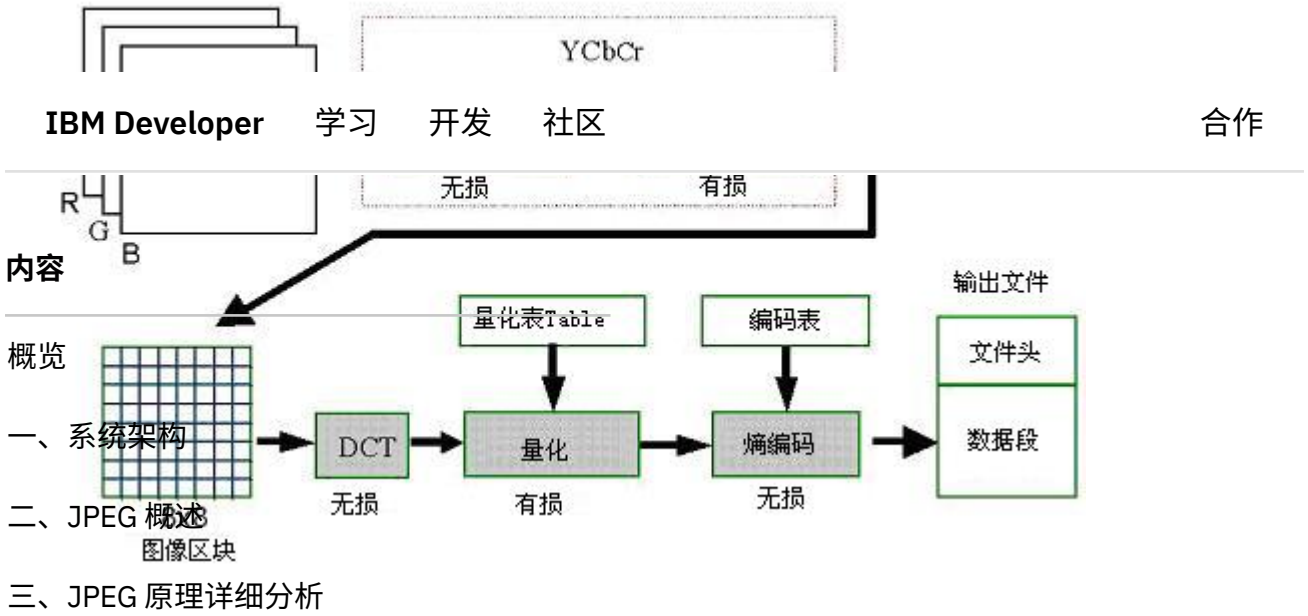
评论3. 重排列 DCT 结果

4. 量化

5. RLE 编码

6. 范式 Huffman 编码

7. DC 的编码



#### 四、JPEG文件存储格式

### 1、色彩空间 color space

#### 五、JPEG 压缩过程的优化

在图像处理中，为了利用人的视角特性，从而降低数据量，通常把 RGB 空间表示的彩色图像转换为 YUV 或 YCrCb 空间表示。

#### 六、JPEG 在本嵌入式 Linux 应用中遇到的问题

现在采用的色彩空间变换有三种：YIQ，YUV 和 YCrCb。

#### 七、总结

每一种色彩空间都产生一种亮度分量信号和两种色度分量信号，而每一种变换使用的相关主题

色彩空间	适用范围
YIQ	NTSC 彩色电视制式
YUV	PAL 和 SECAM 彩色电视制式
YCrCb	计算机用的显示器

YUV 不是哪个英文单词的缩写，而只是符号

表示亮度，UV 用来表示色差，U、V

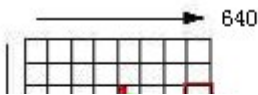
YUV 表示法的重要性是它的亮度信号(Y)和色度信号(U、V)是相互独立的，也就是 Y 信号和 U、V 是独立的，所以可以对这些单色图

信号(U、V)是相互独立的，也就是 Y 信号和 U、V 是独立的，所以可以对这些单色图

举例来说明一下：

要存储 RGB 8：8：8 的彩色图像，即 R、G、B 分量都用 8 位二进制数（1 个字节）表示，需要的存储容量为 640×480×(1+1+1)=900KB，其中(1+1+1)表示

分量都用 8 位二进制数（1 个字节）表示，需要的存储容量为 640×480×(1+1+1)=900KB，其中(1+1+1)表示



内容

如果用 YUV 来表示同一幅彩色图像，Y 分量仍然为640×480，并且 Y 分量仍然用8位值分别用相同的一个值表示，那么存储同样的一幅图像所需的存储空间就减少到640：一、系统要求 20KB。也就是把数据压缩了一半。

二、JPEG 概述

三、JPEG 原理详细分析

四、JPEG 文件存储格式

五、JPEG 压缩过程的优化

六、JPEG 在本嵌入式 Linux 应用中遇到的问题

七、总结 无论是用 YIQ、YUV 和 YCrCb 还是其他模型来表示的彩色图像，由于现在所有的显示每个像素之前，须要把彩色分量值转换成 RGB 值。

相关主题 对于电视机，在考虑人的视觉系统和电视阴极射线管(CRT)的非线性特性之后，RGB 和表示： 评论

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

即：

$$Y=0.3R+0.59G+0.11B$$

$$U=B-Y$$

$$V=R-Y$$



对计算机而言，计算机用的数字域的色彩空间变换与电视模拟域的色彩空间变换不同 RGB 空间的转换关系如下：

IBM Developer

学习    开发    社区

合作

内容

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.299 & 0.578 & 0.114 \\ 0.500 & -0.4187 & -0.0813 \\ -0.1687 & -0.3313 & 0.500 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

从这里，就可以看出，计算出来的 Y、Cr 和 Cb 分量，会出现大量的小数，即浮点数。大量的浮点数的运算，当然经过一定的优化，这些浮点数运算可以用移位与加法这些计

一、系统架构

RGB 与 YCrCb 之间的逆变换关系可写成如下的形式：

二、JPEG 概述

三、JPEG 原理详细分析

四、JPEG 文件存储格式

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 1.4020 & 0 \\ 0 & 0.7141 & -0.3441 \\ 1 & 0 & 1.7720 \end{bmatrix} \begin{bmatrix} 0 \\ Cr - 128 \\ Cb - 128 \end{bmatrix}$$

五、JPEG 压缩过程的优化

总体来说，上面讲的这些内容，主要就是对原始图片，可以先进行色彩空间的处理，

六、JPEG 在本嵌入式 Linux 应用中遇到的问题

请注意，实际上，JPEG 算法与色彩空间无关，色彩空间是涉及到图像采样的问题，

七、总结

因此“RGB 到 YUV 变换”和“YUV 到 RGB 变换”不包含在 JPEG 算法中。JPEG 算法处：它可以压缩来自不同色彩空间的数据，如 RGB，YcbCr 和 CMYK。

相关主题

评论

## 2、色彩深度 color depth

在图像中，它是由很多个点来组成的，那的，从而会使得图片的数据有多和少的区

每个像素点所用的位数就叫做像素深

一幅彩色图像的每个像素用 R，G，B 三24 bit，每个像素可以是2的24次方=16 7

表示，若每个分量用8位，那么一个像5种颜色中的一种。表示一个像素的位

在用二进制数表示彩色图像的像素时，除如，RGB 5：5：5表示一个像素时，用2素深度为16位，而图像深度为15 位。

B 分量用固定位数表示外，往往还增16位表示，其中 R，G，B 各占5位



在用32位表示一个像素时，若 R，G，B 分别用8位表示，剩下的8位常称为 alpha 通道、中断位、属性位。它的用法可用一个预乘 α 通道(premultiplied alpha)的例子说明。

IBM Developer

学习    开发    社区

合作

这个 alpha 值，在这里就用来表示该像素如何产生特技效果。

内容

概览 总体来说，图像的宽高、分辨率越高，就是组成一幅图的像素越多，则图像文件越大和亮度的位数越多，图像文件就越大。

一、系统架构

只有黑白两种颜色的图像称为单色图像(monochrome)，每个像素的像素值用1位存储。二、JPEG 概述 单色图像需要占据37.5 KB的存储空间。

三、JPEG 原理详细分析

而灰度图像，即有色深的黑白图像，如果每个像素的像素值用一个字节表示，而不是四、JPEG文件存储格式 级，每个像素可以是0~255之间的任何一个值，一幅640×480的灰度图像就需要占1

分量。 五、JPEG 压缩过程的优化

六、JPEG 在本嵌入式 Linux 应用中遇到的问题

3、离散余弦变换 DCT

七、总结

将图像从色彩域转换到频率域，常用的变换方法有：

相关主题

评论

傅氏变换
Walsh—Hadamard 沃尔什哈达玛变换
正弦变换
余弦变换——应用最广
斜变换
哈尔变换
K—L 变换

DCT变换的公式为：

$$F(u,v) = \frac{1}{4} C(u)C(v) \left[ \sum_{i=0}^7 \sum_{j=0}^7 f(i,j) \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16} \right]$$

C(u), C(v)=(2)<sup>-1/2</sup>, 当 u, v = 0;  
C(u), C(v)=1, 其他。

$f(i, j)$  经 DCT 变换之后,  $F(0, 0)$  是直流系数, 其他为交流系数。

8x8的原始图像：

内容

概览	[52 55 61 66 70 61 64 73]							
	63 59 55 90 109 85 69 72]							
一、系统架构	62 59 68 113 144 104 66 73]							
	65 68 71 122 154 106 70 69]							
二、JPEG 概述	67 61 68 104 126 88 68 70]							
	75 63 60 70 77 68 58 75]							
三、JPEG 原理详细分析	85 71 64 59 55 61 65 83]							
	87 79 69 68 65 76 78 94]							

四、JPEG 文件存储格式

推移128后, 使其范围变为 -128~127:

五、JPEG 压缩过程的优化

六、JPEG 在本嵌入式 Linux 应用中遇到的问题	[-76 -73 -67 -62 -58 -67 -64 -55]							
	-56 -66 -69 -60 -15 16 -24 -62 -55]							
七、总结	-65 -70 -57 -6 -26 -22 -58 -59]							
	-61 -67 -60 -24 -2 -40 -60 -58]							
相关主题	-49 -63 -68 -58 -51 -60 -70 -53]							
	-43 -57 -64 -69 -73 -67 -63 -45]							
	-41 -49 -59 -60 -63 -52 -50 -34]							

使用离散余弦变换, 并四舍五入取最接近的整数:

	[-415 -30 -61 27 56 -20 -2 0]							
	4 -22 -61 10 13 -7 -9 5]							
	-47 7 77 -25 -29 10 5 -6]							
	-49 12 34 -15 -10 6 2 2]							
	12 -7 -13 -4 -2 2 -3 3]							
	-8 3 2 -6 -2 1 4 2]							
	-1 0 0 -2 -1 -3 4 -1]							
	0 0 -1 -4 -1 0 1 2]							

上图就是将取样块由时间域转换为频率域系数块。

DCT 将原始图像信息块转换成代表不同频率的系数集, 这有两个优点: 其一, 信号集中在少数系数上, 因此可以用很少的比特数; 其二, 频率域分解映射的能量集中在少数系数上, 因此可以用很少的比特数。

合作

## 内容

## 概覽

## 二、JPEG 概述

### 三、JPEG 原理详细分析

#### 四、JPEG文件存储格式

## 4. 量化

题

## 七、总结

## 相关主题

评:

2/3/19, 4:56 PM

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55

IBM Developer

学习

开发

社区

合作

内容

24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

概览

JPEG亮度量化表

一、系统架构	17	18	24	47	99	99	99	99
二、JPEG 概述	16	21	26	66	99	99	99	99
三、JPEG 原理详细分析	24	26	56	99	99	99	99	99
四、JPEG 文件存储格式	49	64	78	87	99	99	99	99
五、JPEG 压缩过程的优化	99	99	99	99	99	99	99	99
六、JPEG 在本嵌入式 Linux 应用中遇到的问题	99	99	99	99	99	99	99	99
七、总结	99	99	99	99	99	99	99	99

JPEG色度量化表

一、系统架构	17	18	24	47	99	99	99	99
二、JPEG 概述	16	21	26	66	99	99	99	99
三、JPEG 原理详细分析	24	26	56	99	99	99	99	99
四、JPEG 文件存储格式	49	64	78	87	99	99	99	99
五、JPEG 压缩过程的优化	99	99	99	99	99	99	99	99
六、JPEG 在本嵌入式 Linux 应用中遇到的问题	99	99	99	99	99	99	99	99
七、总结	99	99	99	99	99	99	99	99

使用这个量化矩阵与前面所得到的 DCT 系数矩阵：

一、系统架构	26	-3	-6	2	2	-1	0	0
二、JPEG 概述	0	-2	-4	1	1	0	0	0
三、JPEG 原理详细分析	2	1	5	-1	-1	0	0	0
四、JPEG 文件存储格式	-4	1	2	-1	0	0	0	0
五、JPEG 压缩过程的优化	1	0	0	0	0	0	0	0
六、JPEG 在本嵌入式 Linux 应用中遇到的问题	0	0	0	0	0	0	0	0
七、总结	0	0	0	0	0	0	0	0

如，使用-415（DC系数）且四舍五入得：

丘的整数

取整

$$\left(\frac{-415}{16}\right) = \text{取整}(-25.9375) = -26$$

总体来说，DCT 变换实际是空间域的低

器。对 Y 分量采用细量化，对 UV 采

量化表是控制 JPEG 压缩比的关键，这个

卓了一些高频量；另一个重要原因是

过程，大量的图像信息被包含在低频率中

量化处理后，在高频率段，将出现大



Run Length Coding，行程编码又称“运行长度编码”或“游程编码”，它是一种无损压缩

IBM Developer

学习 开发 社区

合作

这个数据的一个特点是相同的内容会重复出现很多次，那么就可以用一种简化的方法

内容

(5, 6) (7, 5) (3, 3) (2, 4) (1, 7)

概览

即为它的行程编码。

- 一、系统架构
- 行程编码的位数会远远少于原始字符串的位数。
- 二、JPEG 概述
- 对经过“Z”字形编排过的数据，即可以用行程编码来对其进行大幅度的数据压缩。
- 三、JPEG 原理详细分析

四、JPEG 文件存储格式

我们用一个简单的例子来详细说明一下：

五、JPEG 压缩过程的优化

57, 45, 0, 0, 0, 0, 23, 0, -30, -16, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, .

六、JPEG 在本嵌入式 Linux 应用中遇到的问题

可以表示为

七、总结

(7); (0, 45); (4, 23); (1, -30); (0, -16); (2, 1); EOB

相关主题

即每组数字的头一个表示0的个数，而且为了能更有利于后续的处理，必须是 4 bit，1

评论

码的一个特点。

## 7、范式 Huffman 编码

在直流 DC 系数经过上面的 DPCM 编码，AC 系数经过 RLE 编码后，得到的数据用行程编码来处理。

范式 Huffman 编码即 Canonical Huffman Coding，现在流行的很多压缩方法都使用了 PNG、JPEG、MPEG 等。

对上面的例子中 RLC 后的结果，对它的行程编码，JPEG 里并不直接保存这个数值，这样

数值	组	实际保存值
0	0	-

IBM Developer

学习

开发

社区

合作

内容			110, 111
	-15, .., -8, 8, .., 15	4	0000, .., 0111, 1000, .., 1111
概览	-31, .., -16, 16, .., 31	5	00000, .., 01111, 10000, .., 11111
一、系统架构	-63, .., -32, 32, .., 63	6	.
	-127, .., -64, 64, .., 127	7	.
二、JPEG 概述	-255, .., -128, 128, .., 255	8	.
	256, 256, .., 511	9	.
三、JPEG 原理详细分析	-1023, .., -512, 512, .., 1023	10	.
	-2047, .., -1024, 1024, .., 2047	11	.
四、JPEG 文件存储格式	-4095, .., -2048, 2048, .., 4095	12	.
	-8191, .., -4096, 4096, .., 8191	13	.
五、JPEG 压缩过程的优化	-16383, .., -8192, 8192, .., 16383	14	.
	-32767, .., -16384, 16384, .., 32767	15	.
六、JPEG 在嵌入式 Linux 应用中遇到的问题			

对上面的例子内容，就可以得到：

七、总结

57 为第 6 组的，实际保存值为 111001，编码为 (6, 111001)

相关主题

评论45 编码为 (6, 101101)

23 为 (5, 10111)

-30 为 (5, 00001)

这个时候前面的例子就变为：

(0, 6), 111001 ; (0, 6), 101101 ; (4, 0111; (1, 5), 00001; (0, 4) , 01

这样，括号里的数值正好再合成一个字节 11100101 是前面 0 的个数，低 4 位描述了后面 3 个字节 -32767..32767。

使用上面这个表简化后的内容，再到 Huffman 编码表里去查询，从而得到最后的编码



如06对应 Huffman 表的111000，那么

IBM Developer

学习    开发    社区

合作

21 = (1, 5) --- 11111110110

内容从而得到最后的结果：

概览111000 111001 ; 111000 101101 ; 1111111110011001 10111 ; 1111111

- 一、系统架构
- 使用范式 Huffman 编码表的好处就是使得出现频率高的数字小于8位，而出现频率低
- 二、JPEG概述
- 大大减少数据量。

- 三、JPEG 原理详细分析
- 需要注意的是，在 JPG 文件中，一般有两个 Huffman 表，一个是 DC 用，一个是 AC

四、JPEG文件存储格式

对 DC 编码的部分是单独来处理的，并且是放在上面这个串的最前面。

- 五、JPEG 压缩过程的优化

- 总体来说，到目前为止，我们就得到了最后需要真正存储用的简化后，也即压缩后的
- 六、JPEG 在本嵌入式 Linux 应用中遇到的问题

七、总结

四、JPEG文件存储格式

相关主题

评论介绍了 JPEG 的原理，我们再来结合一个具体的实例来详细讨论上面所涉及到的细节。

我们先来制作一个简单的8X8大小的像素|                      言把它存成JPEG格式。

方法是用 windows 的画图工具，定义一^                      :小的图，用一些色块填充进去，然后

图所示：



IBM Developer

学习 开发 社区

合作

内容

概览

一、系统架构

宽度 (W): 3 高度 (H): 8 默认值 (D)

单位

☐ 英寸 (I) ☐ 厘米 (M) ☒ 像素 (P)

颜色

☐ 黑白 (B) ☒ 彩色 (L)

二、保存成图片文件后缀为 jpg，但按标准来说，它是一种 JFIF 格式标准的文件，里面的图

三、JPEG 原理详细分析  
JFIF 是一个文件格式标准，JPEG 是一个压缩标准，总体来说它们不是一个概念。

四、JPEG文件存储格式

JFIF 是 JPEG File Interchange Format 的缩写，也即 JPEG 文件交换格式。JFIF 是

五、JPEG压缩过程的优化  
图像压缩技术存储摄影图像的方法。JFIF 代表了一种"通用语言"文件格式，它是专门  
输 JPEG 图像而设计的语言

六、JPEG 在嵌入式 Linux 应用中遇到的问题

JFIF 文件格式定义了一些内容是 JPEG 压缩标准未定义的，如 resolution/aspect rat  
七、总结

相关主题

JFIF Segment Format

标记缩写	占用字节	含义	标记值
SOI	2	Start Of Image	FFD8
EOI	2	End Of Image	FFD9
APP0 Marker	2	It's the marker to identify a JPG file with the JFIF specification	FFE0
Identifier	5	Identifier	JFIF(0x4A46494600)
SOF0	2	Start Of Frame	FFC0
SOS	2	Start Of Scan	FFDA
COM	2	Comment	FFFE
DNL	2	Define Number	FFDC
DRI	2	Define Restart	FFDD
DQT	2	Define Quantization table	FFDB
DHT	2	Define Huffman table	FFC4

我们可以打开 JPEG 文件查看里面的内容，即可看到上面的各个标记段：

IBM Developer

学习 开发 社区

合作

内容

概览

一、系统架构

二、JPEG 概述

三、JPEG 原理详细分析

四、JPEG 文件存储格式

五、JPEG 压缩过程的优化

六、JPEG 在嵌入式 Linux 应用中遇到的问题

七、总结

相关主题

评论

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
00000000h	FF	D8	FF	EO	00	10	4A	46	49	46	00	01	01	01	00	60 ; '???.JFIF.....'	
00000010h	00	60	00	00	FF	DB	00	43	00	08	06	06	07	06	05	08 ; '...' ?C.....	
00000020h	07	07	07	09	09	08	0A	0C	14	0D	0C	0B	0B	0C	19	12 ; .....	
00000030h	13	0F	14	1D	1A	1F	1E	1D	1A	1C	1C	20	24	2E	27	20 ; ..... \$.'	
00000040h	22	2C	23	1C	1C	28	37	29	2C	30	31	34	34	34	1F	27 ; ",#..(7 ,01444.'	
00000050h	39	3D	38	32	3C	2E	33	34	32	FF	DB	00	43	01	09	09 ; 9=82<.342 ?C... SOS0:	
00000060h	09	0C	0B	0C	18	0D	0D	18	32	21	1C	21	32	32	32	32 ; .....2!.!2222-Start Of Frame 0	
00000070h	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32 ; 2222222222222222	
00000080h	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32 ; 2222222222222222	
00000090h	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	FF	CO ; 2222222222222222
000000a0h	00	11	08	00	08	00	08	03	01	22	00	02	11	01	03	11 ; .....".	
000000b0h	00	00	00	00	00	00	00	01	05	01	01	01	01	01	01	00 ; . ?.....	
000000c0h	00	00	00	00	00	00	00	01	02	03	04	05	06	07	08	09 ; .. ??.....	
000000d0h	01	05	FF	C4	00	85	10	00	02	01	03	03	02	04	03	05 ; .. ??.....	
000000e0h	05	04	04	00	00	01	7D	01	02	03	00	04	11	05	12	21 ; .....).	
000000f0h	31	41	06	13	51	61	07	22	71	14	32	81	91	A1	08	23 ; 1A..Qa."q.2△?#	
00000100h	15	52	D1	F0	24	33	62	72	82	09	0A	16	17	B	網	R伴\$3br?... 網	
00000110h	18	19	1A	25	26	27	28	29	2A	34	35	36	37	38	39	3A ; ...\$&'( )*456789:	
00000120h	43	44	45	46	47	48	49	4A	53	54	55	56	57	58	59	5A ; CDEFGHIJSTUVWXYZ	
00000130h	63	64	65	66	67	68	69	6A	73	74	75	76	77	78	79	7A ; cdefghijstuvwxyz	
00000140h	83	84	85	86	87	88	89	8A	92	93	94	95	96	97	98	99 ; 傲厚噲嫫振啟耘掩	
00000150h	9A	A2	A3	A4	A5	A6	A7	A8	A9	AA	B2	B3	B4	B5	B6	B7 ; 殺Y'x: 齒吹斗	
00000160h	B8	B9	BA	C2	C3	C4	C5	C6	C7	C8	C9	CA	D2	D3	D4	D5 ; 腹郝嫫牌拆塢矣哉	
00000170h	D6	D7	D8	DA	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	FA	F1 ; 肿到叁仟澳晓恬翠	
00000180h	F2	F3	F4	F5	F6	F7	F8	F9	FA	FF	C4	00	17	01	00	03 ; 嫫嫫  ??...	
00000190h	01	01	01	01	01	01	01	01	01	00	00	00	00	00	00	01 ; .....	
000001a0h	02	03	04	05	06	07	08	09	0A	0B	FF	C4	00	B5	11	00 ; ..... ??.	
000001b0h	02	01	02	04	04	03	04	07	05	04	04	00	01	02	77	00 ; .....v.	
000001c0h	01	02	03	11	04	05	21	31	06	12	41	51	07	61	71	13 ; .....!1..AQ.aq.	
000001d0h	22	32	81	08	14	42	91	A1	B1	C1	09	23	33	52	F0	15 ; "2?.B嫫嫫.#3R?	
000001e0h	62	72	D1	0A	16	24	34	E1	25	F1	17	18				27 ; br?.\$4??...&'	
000001f0h	28	29	2A	35	36	37	38	39	3A	43	44	45				49 ; ()*56789:CDEFGHI	
00000200h	4A	53	54	55	56	57	58	59	5A	63	64	65				69 ; JSTUVWXYZcdefghi	
00000210h	6A	73	74	75	76	77	78	79	7A	82	83	84				88 ; jstuvwxyz們刺啞?	
00000220h	89	8A	92	93	94	95	96	97	98	99	9A	A2				16 ; 嫫振啟耘掩殺Y'x	
00000230h	A7	A8	A9	AA	B2	B3	B4	B5	B6	B7	B8	B9				24 ; x: 齒吹斗腹郝嫫	
00000240h	C5	C6	C7	C8	C9	CA	D2	D3	D4	D5	D6	D7				12 ; 牌拆塢矣哉肿到	
00000250h	E3	E4	E5	E6	E7	E8	E9	EA	F2	F3	F4	F5				19 ; 沅三玷嫫嫫嫫	
00000260h	FA	FF	DA	00	0C	03	01	00	02	11	03	11				18 ; ??.....?.	
00000270h	AC	7C	31	7B	27	88	1D	C3	C7	0E	82	6E				16 ; 瑋1('?.們.倪 瑋	
00000280h	3E	E4	64	3E	6A	38	75	8B	6A	70	10	E7				2C ; >銳>j8u嫫p.鎖嫫,	
00000290h	79	25	36	D1	45	15	B5	3C	55	78	37	C9				33 ; y\$6移..?Ux7? ?	
000002a0h	5A	DC	B3	E5	7C	A9	69	D1	25	E5	F7	BE				1 ; z++嫫 ?嫫嫫 0	

SOS:

Start Of Scan

image

从图上可以看出：

在头部有 FFD8 ，表示图像的开始；结束部分有 FFD9 ，表示图像的开始。

IBM Developer

学习    开发    社区

合作

还有图像大小信息对应的 FFC0

内容再后面有四个 Haffman 表对应的 FFC4 ；

概览一般一个 JPG 文件里会有 2 类 Haffman 表：一个用于 DC 一个用于 AC ，也即实际有一、DC 系统 AC 两个。

二、JPEG 概述然后就是图像数据段标记 FFDA ；

三、JPEG 原理详细分析我们再来看看各个标记的细部，具体分析一下各个部分的含义。

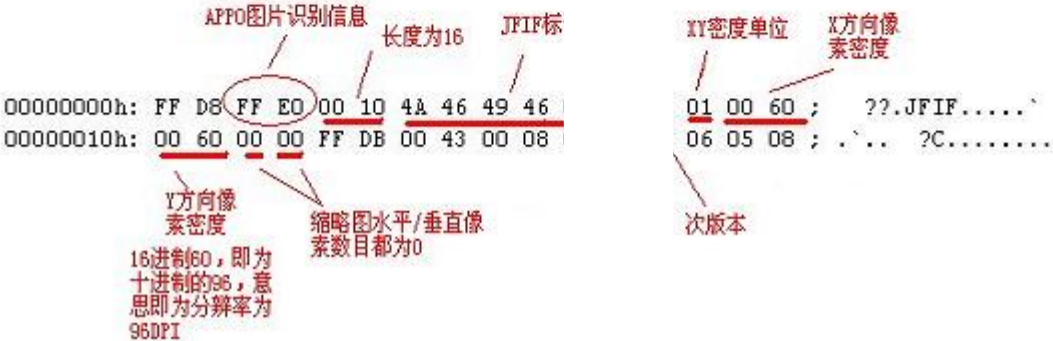
四、JPEG文件存储格式

五、JPEG 压缩过程的优化

1、图片的识别信息

六、JPEG 在本嵌入式 Linux 应用中遇到的问题

结构	总结 (高字节, 低字节), 2 字节
七、	标识符(identifier), 5 字节
相关主题	版本号(version), 2 字节
评论	1 字节主版本号,
	1 字节次版本号
	X 和 Y 的密度单位(units=0: 无单位; units=1: 点数/英寸; units=2: 点数/厘米), 1 字节
	X 方向像素密度(X density), 2 字节
	Y 方向像素密度(Y density), 2 字节
	缩略图水平像素数目(thumbnail horizontal pixels), 1 字节
	缩略图垂直像素数目(thumbnail vertical pixels), 1 字节
	缩略图 RGB 位图(thumbnail RGB bitmap), 由前面的数值
	自 3n, n 为缩略图





上面的内容，在标记 FFE0 后，即为长度16。然后是5字节的 JFIF 标识符号，说明这  
本号码。下一个为 XY 像素的单位，这里为1，表示单位为点数/英寸。然后是 XY 方

内容、量化表的实例

概览

结构

一、系统架构

二、JPEG 概述

三、JPEG 原理详细分析

四、JPEG 文件存储格式

五、JPEG 压缩过程的优化

六、JPEG 在本嵌入式 Linux 应用中遇到的问题

七、总结

长度(高字节, 低字节)
QT 信息 (1 byte): 低 4 位: QT 号(0.3, 否则错误), 高 4 位: QT 精度, 0=8 bit, 否则 16 bit
QT 表的描述表, n = 64*(精度+1)

DQT: Quantization Table

长度: 67

QT 信息: 1 字节  
低 4 位: 0, QT 号  
高 4 位: 0, QT 精度是 8bit

从这里开始后面 64\*(精度+1) 为 QT 表数据, 也即 64 字节

00000010h: 00 60 00 00 FF DB 00 43 00 08 06 06 07 06 05 08 ; .'. ?C.....  
00000020h: 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12 ; .....  
00000030h: 13 0F 14 1D 1A 1F 1E 1D 1A 1C 1C 20 24 2E 27 20 ; ..... \$.!  
00000040h: 12 09 23 10 30 31 34 34 34 1F 27 ; ",#.. (7),01444.  
00000050h: 39 3D 38 32 3C 2E 33 34 32 FF DB 00 43 01 09 09 ; 9=82<.342 ?C...  
00000060h: 09 0C 0B 0C 18 0D 0D 18 32 21 1C 21 32 32 32 32 ; .....2!.!2222  
00000070h: 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 ; 2222222222222222  
00000080h: 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 ; 2222222222222222  
00000090h: 32 32 32 32 32 32 32 32 32 32 32 32 32 FF C0 ; 22222222222222

相关主题

评论

上面这个内容，FFDB 标记后的长度值为(是8bit。然后后面就是64个8x8的 QT 表的

下来的是 QT 信息，占一个字节；这！tem 了。

也即第一个 DQT 量化表的内容表示为十进

8	6	5	8	12	20	26	31
6	6	7	10	13	29	30	28
7	7	8	12	20	29	35	60
7	9	11	15	26	44	50	31
9	11	19	28	34	56	52	39
12	18	28	32	61	52	57	46
25	32	39	57	52	61	60	51
36	46	39	49	56	50	52	50

这个表即为 JPEG 亮度量化表。

内容	9	9	12	50	50	50	50	50
	9	11	24	33	50	50	50	50
概览	12	13	28	50	50	50	50	50
	24	33	50	50	50	50	50	50
	50	50	50	50	50	50	50	50
一、系统架构	50	50	50	50	50	50	50	50
	50	50	50	50	50	50	50	50
	50	50	50	50	50	50	50	50
二、JPEG 概述								

三、这个表的内容即为 JPEG 色度量化表。

四、JPEG 文件存储格式  
当你打开不同的 JPEG 文件，你会看到这两个表可能也是会有区别的。这个主要是使

五、JPEG 压缩过程的优化

六、JPEG 在本嵌入式 Linux 应用中遇到的问题  
3、图像信息段

七、总结

相关主题

评论

结构

长度: (高字节, 低字节)
----------------

IBM Developer

学习 开发 社区

合作

内容

概览

components 数量(1 byte): 灰度图是 1, YCbCr/YIQ 彩色图是 3, CMYK 彩色图是 4
每个 component: 3 bytes:
component id (1 = Y, 2 = Cb, 3 = Cr, 4 = I, 5 = Q)
采样系数 (bit 0-3 vert, 4-7 hor)
量化表编号

一、系统架构

二、JPEG 概述

三、JPEG 原理详细分析

四、JPEG 文件存储格式

五、JPEG 压缩过程的优化

六、JPEG 在本嵌入式 Linux 应用中遇到的问题

七、总结

这个内容，FFC0 标记后即是长度，为17，然后是一个字节的数据精度，通常是：占两字节，这里即为8，然后是图片的宽度，也为8，这也就是我们定义的8x8的内容示 YUV。接下来是三组数据，每组数据里，第一个是 component ID，第二个是采样评论，水平是2。再后面就是量化表的编号了。



## 4、Haffman 表的实例



结构

长度: (高字节, 低字节)

IBM Developer

学习    开发    社区

合作

值表, n bytes: 一个包含了按递增次序, Huffman 编码组号对应的各个值, (n= 代码总数)

内容

概览

一、系统架构

二、JPEG 概述

三、JPEG 原理详细分析

四、JPEG文件存储格式

上面这个内容, FFC4 标记后的内容为数据长度, 再接着的1字节为 Huffman Table 的

五、JPEG 压缩过程的优化  
类型标记, 再高三位是为0。

六、JPEG 在本嵌入式 Linux 应用中遇到的问题  
第一个 DHT 表, 00, 类型为 DC table, HT ID 号为 0;

七、第二个 DHT 表, 10, 类型为 AC table, HT ID 号也为 0;

相关主题  
第三个 DHT 表, 01, 类型为 DC table, HT ID 号为 1;

评论  
第四个 DHT 表, 11, 类型为 AC table, HT ID 号为 1;

即前两个表为Y亮度分量的 DC/AC 表, 后                      UV 色度分量的 DC/AC 表。

以第一个表为例, 因为长度只有 31, 那么                      面的 16 字节, 即绿色部分:

Huffman 编码组号	1	2	3	4	5	6	7	8	0	11	12	13	14	15	16
组中个数	00	01	05	01	01	01	01	01	0	00	00	00	00	00	00

组号为 1 的组中, 代码有 0 个;

组号为 2 的, 代码有 1 个;

组号为 3 的代码有 5 个；

IBM Developer    学习    开发    社区

合作

总共 12 个。

内容  
再看后续的数据：

概览  
00 01 02 03 04 05 06 07 08 09 0A 0B

一、系统架构

即对应：

二、JPEG 概述

三、JPEG 原理详细分析	保存值
2 00	00
四、JPEG 文件存储格式	
3 01 02 03 04 05	001 010 011 100 101
4 06	0110
五、JPEG 压缩过程的优化	
5 07	00111
6 08	001000
六、JPEG 在嵌入式 Linux 应用中遇到的问题	
7 09	0001001
8 0A	00001010
七、总结	
0B	000001011

其他未出现的组号，对应的数据未使用到。也就是说前面提到过的范式 Huffman 编码 8x8 的图像数据很小。

评论

第二个 DHT 表就更复杂些了，长度有 181。

## 5、图像数据段

扫描线中组件的符号 (11-1-1) 通常只 2

合作

bit 4..7: DC table (0..3)

00000260h: FA FF DA 00 0C 03 01 00 02 11 03 11 00 3F 00 E8 ; ??.....?.?  
00000270h: AC 7C 31 7B 27 88 DD C3 C7 0E 82 6E A7 83 EC 6B ; 璋(?'们.肥 踏  
00000280h: 3E E4 64 3E 6A 3B 75 8B 6A 70 10 E7 71 F9 54 2C ; >銃+j8u焯p.鑽錫  
00000290h: 79 25 36 D1 45 15 B5 3C 55 78 37 C9 36 AF D9 B3 ; y6移..?ux?? ?  
000002a0h: 5A DC B3 E5 7C A9 69 D1 25 E5 F7 BE A7 FF D9 ; 2#錯...鑲晶 口

### 三、JPEG 原理详细分析

Start Of Scan	长度: 12	component数量	component ID	使用的 Huffman 表
000000000000	000000000000	000000000000	000000000000	000000000000

Y分量为:      UV分量为:

使用HT ID为0的 DC/AC表

五、这里 SOS 段的长度为 12，后面所含有的 component 数量为 3 个，也即 Y UV。然后用的 Huffman 表的 ID 是多少。

## 六、JPEG 在本嵌入式 Linux 应用中遇到的问题

题 在这个段的后面就是所有压缩后的数据。直到结束的问题，即 FFD9，EOI（End Of I

## 七、总结

## 相关主题

## 五、JPEG 压缩过程的优化

## 评论

JPEG 在目前的应用范围是非常广泛的，许多嵌入式系统中也大量地使用了 JPEG 压缩算法。在这些领域由于传输数据的带宽限制和存贮数据的容量的限制，常常需要使用压缩后的数据，以充分利用带宽和存储空间，达到更好的利用效率。这样，在嵌入式系统中，很需要再对 JPEG 编码压缩的过程进行优化。

## 浮点运算的优化

我们回头查看一下 JPEG 压缩中的 DCT 变换，公式：

$$F(u,v) = \frac{1}{4} C(u)C(v) \left[ \sum_{i=0}^7 \sum_{j=0}^7 f(i,j) \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16} \right]$$

IBM Developer

学习    开发    社区

合作

由于公式中有两个 i/j=0~7 的部分，这样要获得一个 DCT 系数，需要做 8 x 8=64 次乘法和 8 x 8=64 次加法。整个 8x8 像素的 DCT 需要 8 x 8 x 8 x 8=4096 次乘法和 8 x 8 x 8 x 8=4096 次加法。计算量是很大的。

对于有些无浮点运算的嵌入式系统或无专门的数学运算协处理器的系统，会造成大量数据溢出，导致系统崩溃。

一、系统架构  
上面的公式属于 DCT 的二维计算方式，经过简化，可以将其简化为两个一维的公式：

二、JPEG 概述

三、JPEG 原理详细分析  
$$y(u,v) = \frac{1}{\sqrt{2}} \sum_{i=0}^{N-1} c(u) z(v,i) \cos \frac{(2i+1)u\pi}{2N}$$

四、JPEG 文件存储格式  
$$z(v,i) = \frac{1}{\sqrt{2}} \sum_{j=0}^{N-1} c(v) x(i,j) \cos \frac{(2j+1)v\pi}{2N}$$

五、JPEG 压缩过程的优化

六、这样在上面的过程就可以简化为分别计算行和列的 DCT 变换。  
题

对于一行来说需要计算的是 (8 x 8) 次乘法和 (8 x 8) 次加法，8 行就是 8 x (8 x 8) 次乘法和 8 x (8 x 8) 次加法，那么总数就为 2 x (8 x (8 x 8))=1024 次乘法和 2 x (8 x (8 x 8))=1024 次加法，运算量还是比较大的。

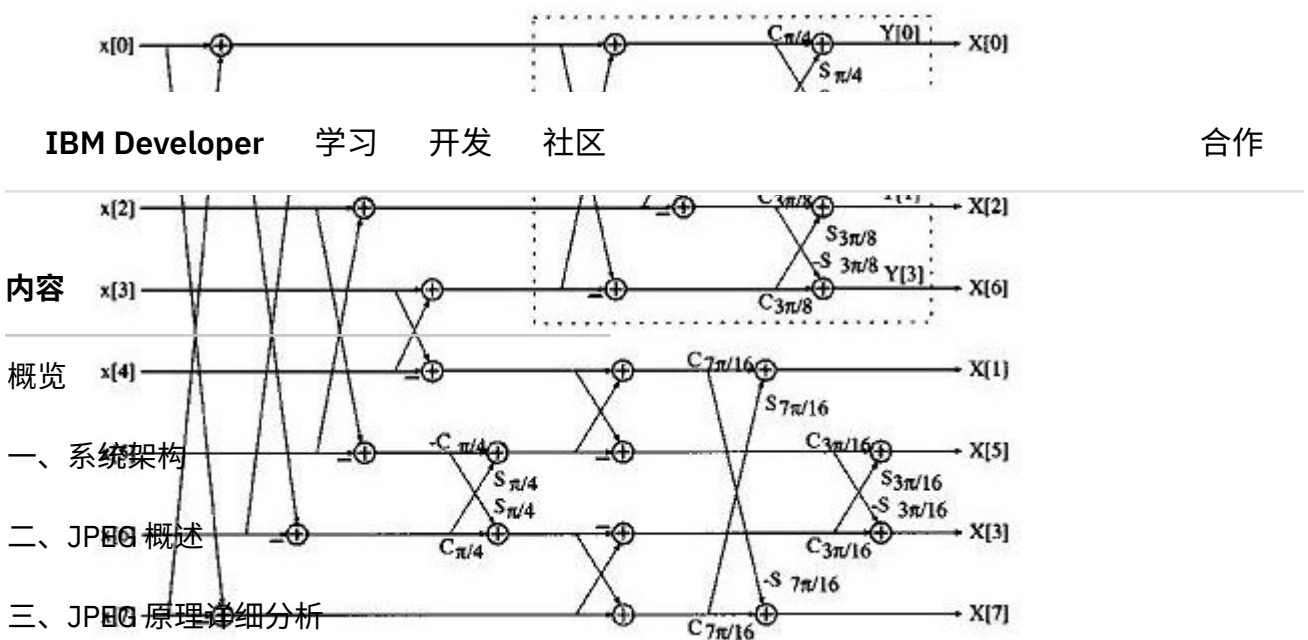
相关主题

但是这样的运算数量还是太大，还需要进一步优化。

评论

在很多嵌入系统中，很多情况下需要不使用浮点运算，这样就需要再找出一维 DCT 的算法。

在对一维 DCT 的运算中，还可以分为奇数数列/行和偶数数列/行。



四、JPEG 的存储格式又出现了多种优化：ChenDCT，LeeDCT，AAN 算法和 LLM 算法。

五、JPEG 压缩过程的优化其中 AAN 算法只需要 29 次加法和 5 次乘法。（注意，它是指每次一维运算要 29 次

法和 5\*8\*2 次乘法的）  
六、JPEG 在嵌入式 Linux 应用中遇到的问题

七、总结

相关主题

评论

$$\begin{bmatrix} Y[0] \\ Y[1] \\ Y[2] \\ Y[3] \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \begin{bmatrix} X[0] \\ X[2] \\ X[4] \\ X[6] \end{bmatrix} + \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ g & -e & d & -b \end{bmatrix} \begin{bmatrix} X[1] \\ X[3] \\ X[5] \\ X[7] \end{bmatrix}$$
$$\begin{bmatrix} Y[7] \\ Y[6] \\ Y[5] \\ Y[4] \end{bmatrix} = \begin{bmatrix} a & c & a & f \\ a & f & -a & -c \\ a & -f & -a & c \\ a & -c & a & -f \end{bmatrix} \begin{bmatrix} X[0] \\ X[2] \\ X[4] \\ X[6] \end{bmatrix} - \begin{bmatrix} b & d & e & g \\ d & -g & -b & -e \\ e & -b & g & d \\ -e & d & -b & g \end{bmatrix} \begin{bmatrix} X[1] \\ X[3] \\ X[5] \\ X[7] \end{bmatrix}$$

其中 Y[0]-Y[7] 都是 1\*8 的矩阵，X[1]-X[7] 是 1\*8 的矩阵。

{a, b, c, d, e, f, g} = 1/2 { cos(pi/4), cos(pi/8), cos(3pi/16), cos(5pi/16), cos(7pi/16), cos(9pi/16), cos(11pi/16), cos(13pi/16) }

再对上面的含有 pi 的系数进行整数优化，避免浮点运算，就会得到：



IBM Developer 学习 开发 社区

合作

内容

概览

一、系统架构

二、JPEG 概述

三、JPEG 原理详细分析

四、JPEG 文件存储格式

其中：

五、JPEG 压缩过程的优化

六、JPEG 在嵌入式 Linux 应用中遇到的问题

$$5/8 = 1/2 + 1/8$$

七、总结

$$7/8 = 1 - 1/8$$

相关主题

上面的除以 2，除以 8，都可以通过移位来实现，即右移一位和右移三位。即总数为

这样就在很大程度上将原本需要使用乘法似的值来完成整个转换过程，会有很好的

运算的过程全部转换成了简单的加法处理效果。

在处理上面的数据中，可以使用一些中间中间值。

记录中间结果，这样就可以减少反复

$$tmp0 = x[0] + x[7];$$

$$tmp7 = x[0] - x[7];$$

$$tmp1 = x[1] + x[6];$$



```
tmp6 = x[1] - x[6];
```

```
tmp5 = x[2] - x[5];
```

内容

```
tmp3 = x[3] + x[4];
```

概览

```
tmp4 = x[3] - x[4];
```

一、系统架构

```
tmp10 = tmp0 + tmp3;
```

二、JPEG 概述

三、JPEG 原理详细分析

```
tmp13 = tmp0 - tmp3;
```

四、JPEG 文件存储格式

```
tmp11 = tmp4 + tmp2;
```

五、JPEG 压缩过程的优化

```
tmp12 = tmp11 - tmp2;
```

六、JPEG 在本嵌入式 Linux 应用中遇到的问题

```
/* 对偶数项进行运算 X 0,4,6,2 */
```

七、总结

```
X[0] = tmp1 + tmp11;
```

相关主题

```
X[4] = tmp10 / 2 - tmp11
```

评论

```
X[6] = tmp12 - (tmp13/4 + tmp13/8);
```

```
X[2] = tmp12/4 + tmp12/8 - tmp13;
```

其他的各个值也是类似处理的。

# 六、JPEG 在本嵌入式 Linux 应用中遇到的问题

在本系统中，提供给用户一些播放图片和保存图片的功能，在这个过程中就需要使用

## 1、JPEG 出错的处理

在对图片做预览处理的时候，有些图片原始尺寸很大，那么就需要将其转换成较小的



了一个问题，即有时需要显示的图片，会导致系统无响应。

IBM Developer学习开发社区合作

在前面的部分，有说到 JPEG 文件的格式中，JPEG 结束的标记 EOI (End Of Image)

内容

如果需要显示的图片，在传输过程中，或转换过程中，出现了没有 EOI 数据，那么应

概览

一、JPEG 解码的效率优化

二、JPEG 概述

在解码 JPEG 时，可以使用 software decode 或 hardware decode 来处理。Hardwa

三、JPEG 原理详细分析

硬件解码功能，其解码速度会较 software decode 有数量级的提高。但有时使用 har

三、JPEG 原理详细分析

提供的 SDK 会是直接访问硬件，将 jpeg 直接输出到显示设备，从而会导致 hardware c

四、JPEG 文件存储格式

而使用 software decode，就能在应用层完全掌握 jpeg decode 的数据缓冲结果，并

五、JPEG 压缩过程的优化

混合处理，从而会有较高的灵活性。并且使用 decode buffer cache，来将已经解码的

六、不必反复去解码 JPEG 图片，从而也能有效提高绘图效率。

六、不必反复去解码 JPEG 图片，从而也能有效提高绘图效率。

问题

七、总结

相关主题

上面的内容是本人对 JPEG 原理做的一个详细的实例分析，还介绍了 JPEG 编码过程

评论

资源有限的嵌入系统中避免大量的浮点运算。

在对 JPEG 原理做了一个详细的分析后，对 JPEG 涉及到的各个细节有了一个

编码过程来分析时，将会有有一个清楚的全

本文结合应用实例，对在嵌入式 Linux 应到到的 JPEG 有关的问题，做了一个说

考。

现在，你如果再回头去看 JPEG 的原理，应该能看懂它整个过程的来龙去脉了。

相关主题

在 [developerWorks Linux 专区](#)，找到更多针对 Linux 开发人员的资源，并阅览我

在 developerWorks 上查阅所有 [Linux 技巧](#) 和 [Linux 教程](#)。

IBM Developer

学习   开发   社区

合作

评论

内容  
添加或订阅评论，请先[登录](#)或[注册](#)。

概览 ☐ 有新评论时提醒我

一、系统架构

IBM Developer

站点反馈

我要投稿

报告滥用

第三方提示

关注微博

大学合作

选择语言

English

中文

日本語

Русский

Português (Brasil)

Español

한글

Code patterns

技术文档库

**IBM Developer**

学习

开发

社区

合作

---

开发者中心

订阅源

时事通讯

视频

博客

活动

社区

联系 IBM

隐私条约

使用条款

信息无障碍选项

反馈

Cookie 首选项