```
侵蚀昨天 ● 于 2018-07-03 16:51:56 发布 ● 18544 🏚 收藏 58
                                       文章标签: G711 audio
                                     本文目的:
                                  1、熟悉G711a/u两种格式的基本原理
                                  2、熟悉两种压缩算法的实现步骤及提供源码实现
                                     它是国际电信联盟ITU-T定制出来的一套语音压缩标准,它代表了对数PCM(logarithmic pulse-code modulation)抽样标准,主要用于电话。它主要用
                                  脉冲编码调制对音频采样,采样率为8k每秒。它利用一个 64Kbps 未压缩通道传输语音讯号。 起 压缩率为1:2, 即把16位 数据压缩成8位。G.711是主流的
                                  波形声音编解码器。
                                     G.711 标准下主要有两种压缩算法。一种是 u-law algorithm (又称often u-law, ulaw, mu-law),主要运用于北美和日本;另一种是 A-law
                                  algorithm,主要运用于欧洲和世界其他地区。其中,后者是特别设计用来方便计算机处理的
                                    G711的内容是将14bit(uLaw)或者13bit(aLaw) <mark>采样<sup>Q</sup></mark>的PCM数据编码成8bit的数据流,播放的时候在将此8bit的数据还原成14bit或者
                                  13bit进行播放,不同于MPEG这种对于整体或者一段数据进行考虑再进行编解码的做法,G711是波形编解码算法,就是一个sample对
                                  应一个编码,所以压缩比固定为:
                                  8/14 = 57\% (uLaw)
                                  8/13 = 62\% (aLaw)
                                  G.711就是语音模拟信号的一种非线性量化, bitrate 是64kbps. 详细的资料可以在ITU 上下到相关的spec,下面主要列出一些性能参数:
                                  G.711 (PCM方式)
                                  • 采样率:8kHz
                                  •信息量:64kbps/channel
                                  • 理论延迟:0.125msec
                                  • 品质: MOS值4.10
                                  算法原理:
                                  A-law的公式如下,一般采用A=87.6
                                  F(x)=	ext{sgn}(x) \left\{ egin{array}{ll} rac{A|x|}{1+\,\ln{(A)}}, & |x|<rac{1}{A} \ rac{1+\,\ln{(A|x|)}}{1+\,\ln{(A)x}}, & rac{1}{A} \leq |x| \leq 1, \end{array} 
ight.
                                  画出图来则是如下图,用x表示输入的采样值,F(x)表示通过A-law变换后的采样值,y是对F(x)进行量化后的采样值。
                                                        0.5
                                  -1.0
                                              -0.5
                                                                     0.5
                                                                                 1.0
                                                 https://dlog.csdn.net/q2519008
                                  由此可见在输入的x为高值的时候,F(x)的变化是缓慢的,有较大范围的x对应的F(x)最终被量化为同一个y,精度较低。相反在低声强区域,也就是x为低值
                                  的时候,F(x)的变化很剧烈,有较少的不同x对应的F(x)被量化为同一个y。意思就是说在声音比较小的区域,精度较高,便于区分,而声音比较大的区域,
                                  精度不是那么高。
                                  对应解码公式(即上面函数的反函数):
                                  F^{-1}(y) = 	ext{sgn}(y) \left\{ egin{array}{ll} rac{|y|(1+	ext{In}(A))}{A}, & |y| < rac{1}{1+	ext{In}(A)} \ rac{\exp(|y|(1+	ext{In}(A))-1)}{A}, & rac{1}{1+	ext{In}(A)} \le |y| < 1, \end{array} 
ight.
                                      G711A(A-LAW)压缩十三折线法
                                      g711a输入的是13位(S16的高13位),这种格式是经过特别设计的,便于数字设备进行快速运算。
                                      1、取符号位并取反得到s
                                      2、获取强度位eee, 获取方法如图所示
                                      3、获取高位样本位wxyz
                                      4、组合为seeewxyz,将seeewxyz逢偶数为取补数,编码完毕
                                      A-law如下表计算,第一列是采样点,共13bit,最高位为符号位。对于前两行,折线斜率均为1/2,跟负半段的相应区域位于同一段折线上,对于3到8
                                  行,斜率分别是1/4到1/128,共6段折线,加上负半段对应的6段折线,总共13段折线,这就是所谓的A-law十三段折线法。
                                       Linear input code | Compressed code | Linear output code
                                           [note 1]
                                                                         [note 2]
                                                       XOR 01010101
                                                      s000abcd
                                                                     s00000000abcd1
                                       s00000000abcdx
                                       s0000001abcdx
                                                      s001abcd
                                                                     s0000001abcd1
                                                      s010abcd
                                       s000001abcdxx
                                                                     s000001abcd10
                                                      s011abcd
                                       s00001abcdxxx
                                                                     s00001abcd100
                                                      s100abcd
                                                                     s0001abcd1000
                                       s0001abcdxxxx
                                                      s101abcd
                                       s001abcdxxxxx
                                                                     s001abcd10000
                                                      s110abcd
                                       s01abcdxxxxxx
                                                                     s01abcd100000
                                       s1abcdxxxxxxx
                                                      s111abcd
                                                                    s1abcd1000000 000
                                      示例:
                                      输入pcm数据为1234, 二进制对应为(0000 0100 1101 0010)
                                     二进制变换下排列组合方式(0000010011010010)
                                    1、获取符号位最高位为0,取反, s=1
                                    2、获取强度位00001, 查表, 编码制应该是eee=011
                                    3、获取高位样本wxyz=0011
                                    4、组合为10110011,逢偶数为取反为11100110,得到E6
                                                                                                                                  登录后复制
                                       1 #define SIGN_BIT
                                                                      /* Sign bit for a A-law byte. */
                                                             (0x80)
                                       2 #define QUANT_MASK (0xf)
                                                                   /* Quantization field mask. */
                                          #define NSEGS
                                                             (8) /* Number of A-law segments. */
                                          #define SEG_SHIFT (4) /* Left shift for segment number. */
                                          #define SEG_MASK (0x70)
                                                                       /* Segment field mask. */
                                          static int seg_aend[8] = \{0x1F, 0x3F, 0x7F, 0xFF,
                                                         0x1FF, 0x3FF, 0x7FF, 0xFFF};
                                          static int seg_uend[8] = \{0x3F, 0x7F, 0xFF, 0x1FF,
                                                         0x3FF, 0x7FF, 0xFFF, 0x1FFF};
                                       9
                                       10
                                       11 static int search(
                                              int val, /* changed from "short" *drago* */
                                        12
                                       13
                                              int * table,
                                              int size) /* changed from "short" *drago* */
                                       14
                                       15 {
                                       16
                                                         /* changed from "short" *drago* */
                                              int i;
                                       17
                                              for (i = 0; i < size; i++) {
                                        18
                                                  if (val <= *table++)</pre>
                                       19
                                       20
                                                      return (i);
                                       21
                                              }
                                              return (size);
                                       22
                                       23 }
                                       24
                                       25 int linear2alaw(int pcm_val) /* 2's complement (16-bit range) */
                                       26
                                                                               /* changed from "short" *drago* */
                                       27 {
                                       28
                                                      mask; /* changed from "short" *drago* */
                                              int
                                       29
                                                             /* changed from "short" *drago* */
                                               int
                                                      seg;
                                                      aval;
                                       30
                                               int
                                       31
                                       32
                                              pcm_val = pcm_val >> 3;//这里右移3位,因为采样值是16bit,而A-law是13bit,存储在高13位上,低3位被舍弃
                                       33
                                       34
                                              if (pcm_val >= 0) {
                                       35
                                                                 /* sign (7th) bit = 1 二进制的11010101*/
                                       36
                                                  mask = 0xD5;
                                              } else {
                                       37
                                       38
                                                                  /* sign bit = 0 二进制的01010101*/
                                                  mask = 0x55;
                                       39
                                                  pcm_val = -pcm_val - 1; //负数转换为正数计算
                                       40
                                       41
                                       42
                                              /* Convert the scaled magnitude to segment number. */
                                       43
                                              seg = search(pcm_val, seg_aend, 8); //查找采样值对应哪一段折线
                                        44
                                       45
                                              /* Combine the sign, segment, and quantization bits. */
                                       46
                                                                /* out of range, return maximum value. */
                                       47
                                               if (seg >= 8)
                                                  return (0x7F ^ mask);
                                       48
                                       49
                                               else {
                                           //以下按照表格第一二列进行处理,低4位是数据,5~7位是指数,最高位是符号
                                       51
                                                  aval = seg << SEG_SHIFT;</pre>
                                                  if (seg < 2)
                                       52
                                       53
                                                      aval |= (pcm_val >> 1) & QUANT_MASK;
                                       54
                                                  else
                                                      aval |= (pcm_val >> seg) & QUANT_MASK;
                                       55
                                       56
                                                  return (aval ^ mask);
                                       57
                                              }
                                       58 }
                                          int alaw2linear(int a_val)
                                                     t; /* changed from "short" *drago* */
                                               int
                                                     seg; /* changed from "short" *drago* */
                                               int
                                              a_val ^= 0x55; //异或操作把mask还原
                                              t = (a_val & QUANT_MASK) << 4;//取低4位,即表中的abcd值,然后左移4位变成abcd0000
                                              seg = ((unsigned)a_val & SEG_MASK) >> SEG_SHIFT; //取中间3位,指数部分
                                       10
                                              switch (seg) {
                                              case 0: //表中第一行,abcd0000 → abcd1000
                                       11
                                       12
                                                  t += 8;
                                       13
                                                  break;
                                        14
                                               case 1: //表中第二行,abcd0000 -> 1abcd1000
                                       15
                                                  t += 0 \times 108;
                                        16
                                                  break;
                                       17
                                               default://表中其他行,abcd0000 -> 1abcd1000 的基础上继续左移(按照表格第二三列进行处理)
                                        18
                                                  t += 0 \times 108;
                                                  t <<= seg - 1;
                                       21
                                              return ((a_val & SIGN_BIT) ? t : -t);
                                       22 }
                                      u-law也叫g711u,使用在北美和日本,输入的是14位,编码算法就是查表,计算出:基础值+平均偏移值 没啥复杂算法,就是基础值+平均偏移值,
                                      \mu-law的公式如下,\mu取值一般为255
                                       F(x) = \operatorname{sgn}(x) rac{\ln(1+\mu|x|)}{\ln(1+\mu)} - 1 \leq x \leq 1
                                       F^{-1}(y) = 	ext{sgn}(y)(1/\mu)((1+\mu)^{|y|}_{	ext{ps}} - 1)_{	ext{gr}} - 1 \le y \le 1
                                      相应的µ-law的计算方法如下表
                                                      Compressed code | Linear output value
                                        Linear input value
                                            [note 1]
                                                                           [note 2]
                                                        XOR 11111111
                                        s00000001abcdx
                                                       s000abcd
                                                                      s000000001abcd1
                                        s0000001abcdxx
                                                       s001abcd
                                                                      s00000001abcd10
                                        s000001abcdxxx
                                                       s010abcd
                                                                      s0000001abcd100
                                                       s011abcd
                                                                      s00001abcd1000
                                        s00001abcdxxxx
                                        s0001abcdxxxxx
                                                       s100abcd
                                                                      s0001abcd10000
                                        s001abcdxxxxxx
                                                       s101abcd
                                                                      s001abcd100000
                                        s01abcdxxxxxxx
                                                       s110abcd
                                                                      s01abcd1000000
                                                                      slabcd10000000 900
                                        s1abcdxxxxxxxx
                                                       s111abcd/
                                         14 bit Binary Linear input code 8 bit Compressed code
                                        +8158 to +4063 in 16 intervals of 256 | 0x80 + interval number
                                       +4062 to +2015 in 16 intervals of 128 | 0x90 + interval number
                                                                  0xA0 + interval number
                                        +2014 to +991 in 16 intervals of 64
                                                                  0xB0 + interval number
                                        +990 to +479 in 16 intervals of 32
                                       +478 to +223 in 16 intervals of 16
                                                                  0xC0 + interval number
                                        +222 to +95 in 16 intervals of 8
                                                                  0xD0 + interval number
                                        +94 to +31 in 16 intervals of 4
                                                                  0xE0 + interval number
                                       +30 to +1 in 15 intervals of 2
                                                                  0xF0 + interval number
                                                                  0xFF
                                                                  0x7F
                                        -31 to -2 in 15 intervals of 2
                                                                  0x70 + interval number
                                        -95 to -32 in 16 intervals of 4
                                                                  0x60 + interval number
                                        -223 to -96 in 16 intervals of 8
                                                                  0x50 + interval number
                                        -479 to -224 in 16 intervals of 16
                                                                  0x40 + interval number
                                                                  0x30 + interval number
                                        -991 to -480 in 16 intervals of 32
                                        -2015 to -992 in 16 intervals of 64
                                                                  0x20 + interval number
                                        -4063 to -2016 in 16 intervals of 128 | 0x10 + interval number
                                        -8159 to -4064 in 16 intervals of 256 | 0x00 + interval number
                                      示例:
                                      输入pcm数据为1234
                                    1、取得范围值, 查表得 +2014 to +991 in 16 intervals of 64
                                    2、得到基础值为0xA0
                                    3、得到间隔数为64
                                    4、得到区间基本值2014
                                    5、当前值1234和区间基本值差异2014-1234=780
                                    6、偏移值=780/间隔数=780/64, 取整得到12
                                    7、输出为0xA0+12=0xAC
                                           #define BIAS
                                                             (0x84)
                                                                        /* Bias for linear code. 线性码偏移值*/
                                           #define CLIP
                                                                 8159
                                                                        //最大量化级数量
                                           int linear2ulaw( int
                                                                pcm_val) /* 2's complement (16-bit range) */
                                               int
                                                      mask;
                                               int
                                                      seg;
                                               int
                                                      uval;
                                              /* Get the sign and the magnitude of the value. */
                                        10
                                              pcm_val = pcm_val >> 2;
                                       11
                                              if (pcm_val < 0) {
                                       12
                                                  pcm_val = -pcm_val;
                                        13
                                        14
                                                  mask = 0x7F;
                                              } else {
                                       15
                                       16
                                                  mask = 0xFF;
                                       17
                                              }
                                                  if ( pcm_val > CLIP ) pcm_val = CLIP; /* clip the magnitude 削波*/
                                        18
                                              pcm_val += (BIAS >> 2);
                                       19
                                       20
                                              /* Convert the scaled magnitude to segment number. */
                                       21
                                       22
                                              seg = search(pcm_val, seg_uend, 8);
                                       23
                                       24
                                       25
                                               * Combine the sign, segment, quantization bits;
                                               * and complement the code word.
                                       26
                                       27
                                               */
                                       28
                                              if (seg >= 8)
                                                             /* out of range, return maximum value. */
                                       29
                                                  return (0x7F ^ mask);
                                               else {
                                       30
                                                  uval = (seg << 4) | ((pcm_val >> (seg + 1)) & 0xF);
                                       31
                                       32
                                                  return (uval ^ mask);
                                       33
                                       34
                                       35 }
                                       36
                                       37 int ulaw2linear( int u_val)
                                       38 {
                                       39
                                              int t;
                                       40
                                              /* Complement to obtain normal u-law value. */
                                       41
                                              u_val = \sim u_val;
                                       42
                                       43
                                       44
                                               /*
                                        45
                                               * Extract and bias the quantization bits. Then
                                       46
                                               * shift up by the segment number and subtract out the bias.
                                       47
                                               */
      侵蚀昨天
                                              t = ((u_val & QUANT_MASK) << 3) + BIAS;</pre>
                                       48
             ▼ 暂无认证
                                              t <<= (u_val & SEG_MASK) >> SEG_SHIFT;
                                        49
                                       50
                         61万+
       3万+
            7100
                                              return ((u_val & SIGN_BIT) ? (BIAS - t) : (t - BIAS));
                                       51
           总排名
      周排名
                   访问
 原创
                         等级
                                       52 }
                         1683
 4368
       226
             340
                   81
                                  和A-law画在同一个坐标轴中就能发现A-law在低强度信号下,精度要稍微高一些。
                   评论
                         收藏
       粉丝
 积分
             获赞
                                   0.9
                                   0.8
                     关注
     私信
                                   0.7
                                   0.6
                                   0.5
                                   0.4
  搜博主文章
                                   0.3
                                   0.2
热门文章
                                   0.1
#、##、__VA_ARGS__和
                                               0.2
                                                        0.4
                                                                 0.6
                                                                          0.8
httpA-law/bullawg.csdn.net/q2519008
【音频】I2S协议时序及使用粗解 ① 68157
                                      以上是两种算法的连续条件下的计算公式,实际应用中,我们确实可以用浮点数计算的方式把F(x)结果计算出来,然后进行量化,但是这样一来计算量
Linux:objdump命令解析 ① 63574
                                  会比较大,实际上对于A-law(A=87.6时),是采用13折线近似的方式来计算的,而\mu-law(\mu=255时)则是15段折线近似的方式。
Linux top命令里面%CPU和cpu(s)的差别
                                      参考:
57377
.svn文件太大解决办法 ① 34761
                                      https://en.wikipedia.org/wiki/A-law_algorithm
                                      https://github.com/quatanium/foscam-ios-sdk/blob/master/g726lib/g711.c
分类专栏
                                      https://www.cnblogs.com/charybdis/p/8848457.html
     V4L2
                           1篇
                                      http://www.21ic.com/evm/audio/201705/721797.htm
     NVIDIA
                           2篇
  DL
                           2篇
      从0开始写makefile
                          2篇
                                  g711编码转换成pcm编码
                                                                                                                                       08-05
  ♥TEXKS TI tda2x/3x系列sdk模块化...
                                  把g711转换成pcm格式详情见博客 https://blog.csdn.net/g0415shenw/article/details/81432854
 每天一个linux命令
                           3篇
                                  语音编解码G.711 G.729
                                                                                                                       1. 语音<mark>编码</mark>分类 (1)波形<mark>编码</mark>: 以逼近声音波形为目标,其代表算法有G.711,其声音清楚度好,语音的自然度高,但是压缩效率比较差,常在32kbps以上...
                                  评论 1 您还未登录,请先 登录 后发表或查看评论
最新评论
__attribute__((weak)) 简介及作用
Mr.CarCar: 语文断句没断好
                                  G711(G711a+g711u)编码原理及代码_夜风的博客_g711a
                                                                                                                                         3-9
                                  G711编码的声音清晰度好,语音自然度高,但压缩效率低,数据量大常在32Kbps以上。常用于电话语音(推荐使用64Kbps),sampling rate为8K,压缩率为2,即把...
NVDLA runtime vp 搭建
全佳轩666: 谢谢回复 vp是docker的 驱动加
                                  G.711编码原理_孤独冰刃的博客
                                                                                                                                        3-30
载的是opendla_2 模型是small版本,请 ....
                                  G.711是国际电信联盟ITU-T定制出来的一套语音压缩标准,它代表了对数PCM(logarithmic pulse-code modulation)抽样标准,是主流的波形声音编解码标准,…
NVDLA runtime vp 搭建
                                  G711音频编码格式
                                                                                                                  大想哥: 这个我还真不知道, 我去试试看 谢
                                     最近工作中遇到了G711a音频编码,之前只用到wav和pcm,特写此文对G711格式了解一番。 文章目录音频保存格式采样和量化G711简介参考来源 ...
谢大佬!
NVDLA runtime vp 搭建
                                  G711编码原理及代码 热门推荐
                                                                                                                             szfhy的博客 ① 3万+
侵蚀昨天: opendla涉及到了hw部分寄存器
                                  G711编码的声音清晰度好,语音自然度高,但压缩效率低,数据量大常在32Kbps以上。常用于电话语音(推荐使用64Kbps),sampling rate为8K,压缩率...
值不一样的,如果你编译full,那么open ...
                                  PCM音频处理小工具:录制/播放/G.711a编解码
                                                                                                                                       12-23
NVDLA runtime vp 搭建
                                  1.工程编译环境:VS2010 2.阅读代码及Doc目录下word文档分析,可熟悉以下知识点: (1) 了解简单工厂模式 (2) 了解PCM、WAV常见数据结构 (3...
大想哥: 我是用master版本编除了nv_small
然后vp也make了一遍,但好像编译得到 ...
                                  音频处理——G711标准详解
                                                                                                                              路人coder ② 2073
                                  目录G711简介G711A算法原理压缩方法举例代码G711U算法原理压缩方法举例代码G711A与G711U对比 G711简介 G711是国际电信联盟ITU-T定制...
您愿意向朋友推荐"博客详情页"吗?
                                  G711编码原理ppt
                                                                                                                                       04-12
                                  G711编码原理ppt,介绍ulaw,alaw编码原理
                                  G711编码原理
                                                                                                                       qq472205482的博客 ① 3777
强烈不推荐 不推荐 一般般 推荐 强烈推荐
                                  在正常的语音通话中,信号幅值的分布并不均匀,小信号出现的概率往往大于大信号出现的概率。G.711 正是利用语音信号的这种特性采用非均匀量化<mark>编</mark>...
                                  音频采样及编解码——LPCM、ADPCM、G711、G726、AAC
                                                                                                                             夜风的博客 ① 1万+
最新文章
                                                 最近再查看hi3516a<mark>音频</mark>资料部分,遇到一些<mark>音频</mark>的专业术语,如LPCM、ADPCM、G711、G726等,故查询了一些资料,对这几...
linux等待队列wait queue
                                  g711官方文档
                                                                                                                                       02-26
一段代码计算余弦距离
                                  音频压缩标准g728的官方文档,可以和我上传的其他压缩标准对比着理解。
v4l2摄像头采集流程及应用程序
                                  A律十三折线法G711编解码介绍
                                                                                                                         jackzhouyu的专栏 ② 6857
                                  G711编码算法理解简介G711国际电信联盟ITU-T定制出来的一套语音压缩标准,主要用于对PCM音频数据编码,将PCM16bit数据压缩为为8Bit,它是主...
               2021年 3篇
2022年 10篇
                                  音频编解码(PCM、G711A、G711U、AAC)理解
                                                                                                                                     1845
2020年 14篇
                2019年 32篇
                                  转载于:https://blog.csdn.net/weixin_37779156/article/details/102813455 PCM整理 简介 PCM:又称脉冲<mark>编码</mark>调制。人耳听到的是模拟信号,PCM是把...
2018年 65篇
                2016年 1篇
                                  G711原理
                                                                                                                       audio_algorithm的博客 @ 859
                                  G.711是国际电信联盟ITU-T定制出来的一套语音压缩标准,它代表了对数PCM(logarithmic pulse-code modulation)抽样标准,是主流的波形声音编解码…
                                  音频编码(PCM、G711A、G711U、AAC)理解
                                                                                                                     weixin_37779156的博客 ② 2万+
                                  ** <mark>音频</mark>的定义 ** <mark>音频</mark>:能被人体感知的声<mark>音频</mark>率,定义为20-20000HZ。声音是通过物体振动产生的声波。是通过介质(空气或固体、液体)传播并能被...
同时运行
                                  音频编解码G.711
                                                                                                                                      o 329
                                  G.711也称为PCM(脉冲<mark>编码</mark>调制),是国际电信联盟订定出来的一套语音压缩标准,主要用于电话。PCM:脉冲<mark>编码</mark>调制(Pulse Code Modulation)。早...
Windows
                                  pcm 转 G711a/u 最新发布
                                                                                                                        qq 46055701的博客 ① 423
                                  pcm 2 g711a(输入13位编成8位) 1.将pcm二进制 转成 排列:x xx...x1 xxxx xxxx... 2.符号位取反 3.根据从高位起,第一个 1 的位数 查表 4.wxyz为 第一个 ...
Mac 应用程
                                  g.711 协议 A律和U律的详细规范
                                                                                                                                       04-22
                                  语音脉冲编码PCM协议,是语音编码的初级协议,由此协议模拟到数字才真正开始发展。
                                  G.711编码原理及代码
                                                                                                                          guo8113的专栏 ① 7245
                                  G711音频编码原理及代码。
Mac与Window融合。在Mac
                                                                            "相关推荐"对你有帮助么?
上运行您喜爱的Windows
                                                                                    ∵ 一般 ∵ 有帮助
软件。Mac用户的最优之选。
充分利用Mac。立即试用!
                                                               ◎ 2022 CSDN 皮肤主题:大白 设计师: CSDN官方博客 返回首页
Parallels.cn
                                                关于我们 招贤纳士 商务合作 寻求报道 ☎ 400-660-0108 ☑ kefu@csdn.net ⑤ 在线客服 工作时间 8:30-22:00
```

6

 $\overline{}$

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心

家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 ©1999-2022北京创新乐知网络技术有限公司 版权与免责声明 版权申诉

侵蚀昨天

下载

插件 认证 开源

audio

Q 搜索

登录/注册 会员中心 🎁 足迹 动态