曦云®系列通用计算 GPU

mcJPEG 编程指南

CSPG-23011-020-F3_V02　　2024-03-15

沐曦专有和三级保密信息　　本文档受 NDA 管控

更新记录

版本	日期	更新说明
V02	2024-03-15	新增曦云®系列 GPU 产品信息
V01	2023-12-29	正式版本首次发布

目录

表目录

1. 概述

本文档主要用于指导用户如何在曦云®系列 GPU 上进行 mcJPEG 解码和编码。

1.1 mcJPEG 解码器

mcJPEG 库提供了面向深度学习和超大规模多媒体应用常用图像格式的高性能、GPU 加速的 JPEG 解码功能。可有效利用可用的 GPU 资源以实现最佳性能。

mcJPEG 库支持以下功能：

- 使用 JPEG 图像数据流作为输入
- 从数据流中解析出图像的宽度和高度，并使用获取到的信息管理 GPU 内存分配和解码
- 提供了专用的 API，用于从原始 JPEG 图像数据流中获取图像信息

JPEG 选项

mcJPEG 库支持：

- Baseline 和 Progressive JPEG 编解码
- 针对 3 个颜色通道 Y，Cb，Cr（Y，U，V），支持以下色度子采样：4:4:4；4:2:2；4:2:0；4:4:0；4:1:1；4:1:0

特性

mcJPEG 库支持：

- 在支持的平台上进行 Baseline JPEG 编解码的硬件加速
- 解码时，输入数据位于主机内存中，输出位于 GPU 内存中
- 色彩空间转换
- 支持用户提供内存管理接口进行设备内存和主机固页内存（pinned host memory）分配

1.2 mcJPEG 编码器

通过 mcJPEG 库的编码函数，可以对用户的图像数据进行 GPU 加速压缩，并将其转换为 JPEG 比特流。支持多种格式和色彩空间的输入数据，且可使用参数控制编码过程。

在调用编码函数之前，用户应该使用 3.3 mcJPEG 编码 API 中的辅助函数执行一些先决步骤。

1.3 多 GPU 支持

mcJPEG 的状态和句柄是与在创建时设置的当前设备绑定的。将这些状态和句柄与另一个当前设备一起使用是未定义行为。用户负责跟踪当前设备。

2. JPEG 解码

2.1 使用 JPEG 解码

要进行解码，需提供数据大小和指向文件数据的指针。解码后的图像会放在输出缓冲区中。

要使用 mcJPEG 库，请先调用初始化的辅助函数。

1. 使用辅助函数 `mcjpegCreateSimple()` 或 `mcjpegCreateEx()` 创建 mcJPEG 库句柄。
2. 使用辅助函数 `mcjpegJpegStateCreate()` 创建 JPEG 状态。

   mcJPEG 库中提供以下辅助函数：

   - `mcjpegStatus_t mcjpegGetProperty(libraryPropertyType type, int *value);`
   - `mcjpegStatus_t mcjpegCreate(mcjpegBackend_t backend, mcjpegHandle_t *handle, mcjpeg_dev_allocator allocator);`
   - `mcjpegStatus_t mcjpegCreateSimple(mcjpegHandle_t *handle);`
   - `mcjpegStatus_t mcjpegCreateEx(mcjpegBackend_t backend, mcjpegDevAllocator_t *dev_allocator, mcjpegPinnedAllocator_t *pinned_allocator, unsigned int flags, mcjpegHandle_t *handle);`
   - `mcjpegStatus_t mcjpegDestroy(mcjpegHandle_t handle);`
   - `mcjpegStatus_t mcjpegJpegStateCreate(mcjpegHandle_t handle, mcjpegJpegState_t *jpeg_handle);`
   - `mcjpegStatus_t mcjpegJpegStateDestroy(mcjpegJpegState handle);`

   其他辅助函数，如 `mcjpegSet*()` 和 `mcjpegGet*()`，可用于按句柄配置库功能。更多信息，参见 2.3 mcJPEG 解码 API。

3. 使用 `mcjpegGetImageInfo()` 函数解析 JPEG 图像的宽度和高度。

   以下为 `mcjpegGetImageInfo()` 函数的签名：

```
mcjpegStatus_t mcjpegGetImageInfo(
mcjpegHandle_t           handle,
const unsigned char       *data,
size_t                   length,
int                     *nComponents,
mcjpegChromaSubsampling_t   *subsampling,
int                     *widths,
int                     *heights);
```

   对于要解码的每个图像，将 JPEG 数据指针和数据长度传递给上述函数。`mcjpegGetImageInfo()` 函数是线程安全的。

4. `mcjpegGetImageInfo()`函数的输出之一是 `mcjpegChromaSubsampling_t`，该参数为枚举类型，其枚举列表由从 JPEG 图像解析出的色度子采样属性组成。更多信息，参见 2.2.8 mcJPEG 色度子采样。

5. 使用 mcJPEG 库中的 `mcjpegDecode()`函数对 JPEG 图像进行解码。此函数的签名如下所示：

```
mcjpegStatus_t mcjpegDecode(
mcjpegHandle_t         handle,
mcjpegJpegState_t      jpeg_handle,
const unsigned char    *data,
size_t                 length,
mcjpegOutputFormat_t   output_format,
mcjpegImage_t          *destination,
mcStream_t             stream);
```

在上述 `mcjpegDecode()`函数中，可以使用参数 `mcjpegOutputFormat_t`，`mcjpegImage_t` 和 `mcStream_t` 来设置 `mcjpegDecode()`函数的输出行为。可通过 `mcStream_t` 参数指示提交异步任务的流。

6. 设置 `mcjpegOutputFormat_t` 参数。

`mcjpegOutputFormat_t` 的参数设置可参见下表：

表 2-1 mcjpegOutputFormat_t 的参数描述

Output_format	含义
MCJPEG_OUTPUT_UNCHANGED	返回编码 JPEG 的原始图像格式，不做任何转换
MCJPEG_OUTPUT_RGB	转换为 planar RGB
MCJPEG_OUTPUT_BGR	转换为 planar BGR
MCJPEG_OUTPUT_RGBI	转换为交错排布的 RGB
MCJPEG_OUTPUT_BGRI	转换为交错排布的 BGR
MCJPEG_OUTPUT_Y	仅返回 Y 分量
MCJPEG_OUTPUT_YUV	返回 YUV planar 格式

7. 如上所述，`mcjpegGetImageInfo()`函数的一个重要优点是能利用从输入的 JPEG 图像解析到的图像信息为解码操作分配适当的 GPU 内存。

`mcjpegGetImageInfo()`函数返回宽度、高度和 nComponents 参数。

```
mcjpegStatus_t mcjpegGetImageInfo(
mcjpegHandle_t           handle,
const unsigned char       *data,
size_t                   length,
int                      *nComponents,
mcjpegChromaSubsampling_t  *subsampling,
int                      *widths,
```

```
int                     *heights);
```

用户可以使用获取到的参数：宽度，高度和 nComponents，来计算输出缓冲区所需的大小。

8. 确保 mcjpegImage_t 结构体中填充了已分配缓冲区的指针和行距（pitch）。包含输出指针的 mcjpegImage_t 结构体定义如下：

```
typedef struct
{
   unsigned char * channel[MCJPEG_MAX_COMPONENT];
   size_t pitch[MCJPEG_MAX_COMPONENT];
} mcjpegImage_t;
```

JPEG 库在当前版本中支持的最大分量数为 MCJPEG_MAX_COMPONENT。

9. 最后，使用上述参数调用 mcjpegDecode()函数时，mcjpegDecode()函数将解码数据填充到输出缓冲区。

2.2 mcJPEG 类型声明

2.2.1 mcJPEG 解码后端

```
typedef enum {
    MCJPEG_BACKEND_DEFAULT = 0,
    MCJPEG_BACKEND_HYBRID = 1,
    MCJPEG_BACKEND_GPU_HYBRID = 2,
    MCJPEG_BACKEND_HARDWARE = 3,
    MCJPEG_BACKEND_GPU_HYBRID_DEVICE = 4,
    MCJPEG_BACKEND_HARDWARE_DEVICE = 5
} mcjpegBackend_t;
```

mcjpegBackend_t 枚举用于选择解码后端。

说明

目前仅支持 MCJPEG_BACKEND_DEFAULT。

2.2.2 mcJPEG 主机固页内存分配器接口

```
typedef int  (*tPinnedMalloc)(void**, size_t, unsigned int flags);
typedef int  (*tPinnedFree)(void*);
typedef struct {
    tPinnedMalloc pinned_malloc;
    tPinnedFree pinned_free;
} mcjpegPinnedAllocator_t;
```

当在 mcjpegCreateEx()函数中将 mcjpegPinnedAllocator_t *allocator 参数设置为指向上述 mcjpegPinnedAllocator_t 结构体的指针时，此结构体将用于分配和释放主机固页内存，以便将数

据复制到设备或从设备复制数据。内存分配和内存释放函数的函数原型类似于 mcMallocHost()和 mcFreeHost()函数。如果成功，它们将返回 0，否则将返回非零。

但是，如果在 mcjpegCreateEx()函数中将 mcjpegPinnedAllocator_t *allocator 参数设置为 NULL，则将使用默认的内存分配函数 mcMallocHost()和 mcFreeHost()。使用 mcjpegCreate()或 mcjpegCreateSimple()函数创建库句柄时，将使用默认的内存分配器。

2.2.3 mcJPEG 图像

```
typedef struct {
    unsigned char * channel[MCJPEG_MAX_COMPONENT];
    size_t pitch[MCJPEG_MAX_COMPONENT];
} mcjpegImage_t;
```

mcjpegImage_t 结构体用于填充已分配缓冲区的指针和行距。mcjpegImage_t 结构体包含输出指针。

MCJPEG_MAX_COMPONENT：mcJPEG 库支持的最大分量数。

2.2.4 mcJPEG 设备内存分配器接口

```
typedef int (*tDevMalloc)(void**, size_t);
typedef int (*tDevFree)(void*);
typedef struct {
    tDevMalloc dev_malloc;
    tDevFree dev_free;
} mcjpegDevAllocator_t;
```

用户可以指定库使用自己的设备内存分配器。内存分配和内存释放函数的函数原型类似于 mcMalloc()和 mcFree()函数。如果成功，它们将返回 0，否则将返回非零。应该为 mcjpegCreate()函数提供一个指向具有正确填充字段的 mcjpegDevAllocator_t 结构体的指针。也可设置为 NULL，在这种情况下，将使用默认的内存分配函数 mcMalloc()和 mcFree()。

当在 mcjpegCreate()或 mcjpegCreateEx()函数中将 mcjpegDevAllocator_t *allocator 参数设置为指向上述 mcjpegDevAllocator_t 结构体的指针时，此结构体用于分配和释放设备内存。

但是，如果在 mcjpegCreate()或 mcjpegCreateEx()函数中将 mcjpegDevAllocator_t *allocator 参数设置为 NULL，则将使用默认的内存分配函数 mcMalloc()和 mcFree()。使用 mcjpegCreateSimple()函数创建库句柄时，将使用默认的设备内存分配器。

2.2.5 mcJPEG 解码状态句柄

```
struct mcjpegJpegState;
typedef struct mcjpegJpegState* mcjpegJpegState_t;
```

mcjpegJpegState 结构体存储临时 JPEG 信息。在使用之前，应先初始化。此 JPEG 状态句柄可以重复使用。

2.2.6 mcJPEG 库句柄

```
struct mcjpegHandle;
typedef struct mcjpegHandle* mcjpegHandle_t;
```

库句柄在使用前应先初始化，库句柄是线程安全的，可由多个线程同时使用。

2.2.7 mcJPEG API 返回码

mcJPEG API 遵循以下返回码及其指示符：

```
typedef enum {
MCJPEG_STATUS_SUCCESS = 0,
MCJPEG_STATUS_NOT_INITIALIZED = 1,
MCJPEG_STATUS_INVALID_PARAMETER = 2,
MCJPEG_STATUS_BAD_JPEG = 3,
MCJPEG_STATUS_JPEG_NOT_SUPPORTED = 4,
MCJPEG_STATUS_ALLOCATOR_FAILURE = 5,
MCJPEG_STATUS_EXECUTION_FAILED = 6,
MCJPEG_STATUS_ARCH_MISMATCH = 7,
MCJPEG_STATUS_INTERNAL_ERROR=8,
MCJPEG_STATUS_IMPLEMENTATION_NOT_SUPPORTED = 9
} mcjpegStatus_t;
```

表 2-2 错误码描述

返回的错误(返回码)	描述
MCJPEG_STATUS_SUCCESS (0)	API 调用已成功完成。请注意，许多调用是异步的，且某些错误可能只在同步后才会出现
MCJPEG_STATUS_NOT_INITIALIZED (1)	库句柄未初始化
MCJPEG_STATUS_INVALID_PARAMETER (2)	传递了错误的参数。例如，输入数据为空指针
MCJPEG_STATUS_BAD_JPEG (3)	无法解析 JPEG 流。需确认编码的 JPEG 流及其大小参数是否正确
MCJPEG_STATUS_JPEG_NOT_SUPPORTED (4)	尝试对 mcJPEG 库不支持的 JPEG 流进行解码
MCJPEG_STATUS_ALLOCATOR_FAILURE (5)	内存分配失败
MCJPEG_STATUS_EXECUTION_FAILED (6)	执行出错
MCJPEG_STATUS_ARCH_MISMATCH (7)	设备不足以满足提供的输入参数集（后端，编码流参数，输出格式等输入参数）
MCJPEG_STATUS_INTERNAL_ERROR (8)	执行设备任务时出错
MCJPEG_STATUS_IMPLEMENTATION_NOT_SUPPORTED (9)	功能尚未实现
MCJPEG_STATUS_INCOMPLETE_BITSTREAM (10)	比特流输入数据不完整

2.2.8 mcJPEG 色度子采样

`mcjpegGetImageInfo()` API 的输出之一是 `mcjpegChromaSubsampling_t`。该参数为枚举类型，其枚举列表由从编码的 JPEG 图像中解析到的色度子采样属性组成。`mcjpegGetImageInfo()`函数目前支持以下色度子采样类型：

```
typedef enum {
MCJPEG_CSS_444,
MCJPEG_CSS_422,
MCJPEG_CSS_420,
MCJPEG_CSS_440,
MCJPEG_CSS_411,
MCJPEG_CSS_410,
MCJPEG_CSS_GRAY,
MCJPEG_CSS_410V,
MCJPEG_CSS_UNKNOWN
} mcjpegChromaSubsampling_t;
```

2.3 mcJPEG 解码 API

2.3.1 mcjpegGetProperty()

获取 mcJPEG 库的主版本号、次版本号或补丁级别的数值。

签名

```
mcjpegStatus_t mcjpegGetProperty(
    libraryPropertyType   type,
    int                   *value);
```

参数

参数	输入/输出	内存	描述
libraryPropertyType type	输入	主机	支持的 `libraryPropertyType` 值之一，即 MAJOR_VERSION、MINOR_VERSION 或 PATCH_LEVEL
int *value	输出	主机	请求的特定 `libraryPropertyType` 所对应的数值

返回值

`mcjpegStatus_t` - 错误码，详情参见 2.2.7 mcJPEG API 返回码。

2.3.2 mcjpegGetMacartProperty()

获取用于构建 mcJPEG 库的 MXMACA 软件包的主版本号、次版本号或补丁级别的数值。有关 mcJPEG 库的信息，请参见 2.3.1 mcjpegGetProperty()。

签名

```
mcjpegStatus_t mcjpegGetMacartProperty(
    libraryPropertyType   type,
    int                   *value);
```

参数

参数	输入/输出	内存	描述
libraryPropertyType type	输入	主机	支持的 libraryPropertyType 值之一，即 MAJOR_VERSION、MINOR_VERSION 或 PATCH_LEVEL
int *value	输出	主机	请求的特定 libraryPropertyType 所对应的数值

返回值

mcjpegStatus_t - 错误码，详情参见 2.2.7 mcJPEG API 返回码。

2.3.3 mcjpegCreateSimple()

使用库默认的编解码器实现和内存分配器分配和初始化库句柄。

签名

```
mcjpegStatus_t mcjpegCreateSimple(mcjpegHandle_t *handle);
```

参数

参数	输入/输出	内存	描述
mcjpegHandle_t *handle	输入/输出	主机	库句柄

返回值

mcjpegStatus_t - 错误码，详情参见 2.2.7 mcJPEG API 返回码。

2.3.4 mcjpegCreateEx()

使用提供的参数分配和初始化库句柄。

签名

```
mcjpegStatus_t mcjpegCreateEx(
mcjpegBackend_t backend,
mcjpegDevAllocator_t *dev_allocator,
mcjpegPinnedAllocator_t *pinned_allocator,
unsigned int flags,
mcjpegHandle_t *handle);
```

参数

参数	输入/输出	内存	描述
mcjpegBackend_t backend	输入	主机	后端参数。如果为 DEFAULT，则将自动选择其中一个底层算法
mcjpegDevAllocator_t *dev_allocator	输入	主机	设备内存分配器。`mcjpegDevAllocator_t` 结构体说明，参见 2.2.4 mcJPEG 设备内存分配器接口。如果为 NULL，则将使用默认的 MXMACA 运行时函数 `mcMalloc()`和 `mcFree()`
mcjpegPinnedAllocator_t *pinned_allocator	输入	主机	主机固页内存分配器。`mcjpegPinnedAllocator_t` 结构体说明，参见 2.2.2 mcJPEG 主机固页内存分配器接口。如果为 NULL，则将使用默认的 MXMACA 运行时函数 `mcMallocHost()`和 `mcFreeHost()`
unsigned int flags	输入	主机	初始化库时，可以附加额外的控制标志。当前仅支持 MCJPEG_FLAGS_DEFAULT
mcjpegHandle_t *handle	输入/输出	主机	库句柄

返回值

`mcjpegStatus_t` - 错误码，详情参见 2.2.7 mcJPEG API 返回码。

2.3.5 mcjpegDestroy()

销毁库句柄

签名

```
mcjpegStatus_t mcjpegDestroy(mcjpegHandle_t handle);
```

参数

参数	输入/输出	内存	描述
mcjpegHandle_t handle	输入/输出	主机	要销毁的库句柄

返回值

`mcjpegStatus_t` - 错误码，详情参见 2.2.7 mcJPEG API 返回码。

2.3.6 mcjpegGetHardwareDecoderInfo()

获取硬件解码器详细信息，如引擎数量和每个引擎中可用的核心数量。

签名

```
mcjpegStatus_t mcjpegGetHardwareDecoderInfo(
mcjpegHandle_t handle,
```

```
unsigned int* num_engines,
unsigned int* num_cores_per_engine);
```

参数

参数	输入/输出	内存	描述
mcjpegHandle_t handle	输入	主机	库句柄
unsigned int* num_engines	输入/输出	主机	可用于解码的引擎数量。返回值 0 表示硬件解码器不可用
unsigned int* num_cores_per_engine	输入/输出	主机	每个引擎的核心数量。返回值 0 表示硬件解码器不可用

返回值

`mcjpegStatus_t` - 错误码，详情参见 2.2.7 mcJPEG API 返回码。

2.3.7 mcjpegJpegStateCreate()

分配和初始化 JPEG 处理所需的内部结构体。

签名

```
mcjpegStatus_t mcjpegJpegStateCreate(
mcjpegHandle_t      handle,
mcjpegJpegState_t   *jpeg_handle);
```

参数

参数	输入/输出	内存	描述
mcjpegHandle_t handle	输入	主机	库句柄
mcjpegJpegState_t *jpeg_handle	输入/输出	主机	解码状态句柄

返回值

`mcjpegStatus_t` - 错误码，详情参见 2.2.7 mcJPEG API 返回码。

2.3.8 mcjpegJpegStateDestroy()

销毁解码状态内部结构。

签名

```
mcjpegStatus_t mcjpegJpegStateDestroy(mcjpegJpegState handle);
```

参数

参数	输入/输出	内存	描述
mcjpegJpegState handle	输入/输出	主机	解码状态句柄

返回值

`mcjpegStatus_t` - 错误码，详情参见 2.2.7 mcJPEG API 返回码。

2.3.9 mcjpegGetImageInfo()

解析 JPEG 头部，获取图像的基本信息。

签名

```
mcjpegStatus_t mcjpegGetImageInfo(
mcjpegHandle_t            handle,
const unsigned char        *data,
size_t                    length,
int                      *nComponents,
mcjpegChromaSubsampling_t   *subsampling,
int                      *widths,
int                      *heights);
```

参数

参数	输入/输出	内存	描述
mcjpegHandle_t handle	输入	主机	库句柄
const unsigned char *data	输入	主机	此指针指向编码数据
size_t length	输入	主机	编码数据的大小，以字节为单位
int *nComponents	输出	主机	图像的 component 数目
int *widths	输出	主机	此指针指向数组第一个元素，数组大小为 MCJPEG_MAX_COMPONENT，其中将保存每个通道的宽度（最多 MCJPEG_MAX_COMPONENT 个）。如果通道未编码，则相应的值为零
int *heights	输出	主机	此指针指向数组第一个元素，数组大小为 MCJPEG_MAX_COMPONENT，其中将保存每个通道的高度（最多 MCJPEG_MAX_COMPONENT 个）。如果通道未编码，则相应的值为零

返回值

`mcjpegStatus_t` - 错误码，详情参见 2.2.7 mcJPEG API 返回码。

2.4 mcJPEG 解码示例

```
mcSetDevice(0);
```

```
//创建 mcjpeg 的库句柄
mcjpegHandle_t mcjpegHandle;
mcjpegCreateSimple(&mcjpegHandle);

//创建 mcjpeg 解码的状态
mcjpegJpegState_t mcjpegState;
mcjpegJpegStateCreate(mcjpegHandle,&mcjpegState);

//获取要解码图像的基本信息
//假设 data 指针指向要解码的图像数据
//size 是数据长度
int widths[MCJPEG_MAX_COMPONENT];
int heights[MCJPEG_MAX_COMPONENT];
int channels;
mcjpegChromaSubsampling_t subsampling;
mcjpegGetImageInfo(mcjpegHandle,data,size,&channels,&subsampling,widths,heig
hts);

//分配用以保存解码数据的设备内存
unsigned char * pBuffer = NULL;
mcMalloc(&pBuffer, widths[0] * heights[0] * MCJPEG_MAX_COMPONENT);

mcjpegImage_t imgdesc =
{
{
   pBuffer,
   pBuffer + widths[0]*heights[0],
   pBuffer + widths[0]*heights[0]*2,
   pBuffer + widths[0]*heights[0]*3
  },
  {
   (unsigned int)widths[0],
   (unsigned int)widths[0],
   (unsigned int)widths[0],
   (unsigned int)widths[0]
  }
};

//创建用于解码的流
mcStream_t stream;
mcStreamCreateWithFlags(&stream,mcStreamNonBlocking);

//进行解码
```

```
mcjpegDecode(mcjpegHandle,mcjpegState,data,size,MCJPEG_OUTPUT_YUV,&imgdesc,stream);
mcStreamSynchronize(stream);

//到现在为止，我们在 imgdesc 结构体中获取解码的图像数据
//在此处添加代码以使用图像数据

//不要忘记释放资源
mcFree(pBuffer);
mcjpegJpegStateDestroy(mcjpegState);
mcjpegDestroy(mcjpegHandle);
mcStreamDestroy(stream);
```

3. JPEG 编码

3.1 使用编码器

在调用 mcJPEG 编码函数之前，用户应执行以下先决步骤。

3.1.1 编码参数配置

用户应使用 `mcjpegEncoderParamsCreate()` 函数创建编码参数结构体。该函数将使用默认参数进行初始化。用户可以使用 `mcjpegEncoderParamsSet*()` 函数来设置特定参数。

可以使用 `mcjpegEncoderParamsSetQuality()` 函数将编码质量参数设置为 1 到 100 之间的整数，该编码质量参数作为生成 JPEG 量化表的依据。

应将参数结构体传递给压缩函数。

3.1.2 创建编码状态结构

用户应使用 `mcjpegEncoderStateCreate()` 函数创建编码状态结构体。此函数将为编码过程保留中间缓冲区。应将此状态传递给压缩函数。

3.1.3 对图像进行编码

mcJPEG 库提供了几个接口，用于以不同的格式和色彩空间压缩图像。

3.1.3.1 mcjpegEncodeYUV

此函数的输入为 YUV 色彩空间的图像。应将相应的 YUV 平面数据填充到 `source` 参数中。`chroma_subsampling` 参数表示输入数据的色度子采样。如果编码参数中的色度子采样与输入色度子采样相同，用户的输入数据将直接用于 JPEG 压缩。否则，将对色度进行重采样，以匹配编码参数的色度子采样。

应根据采样因子提供输入数据。也就是说，色度图像平面的大小应与相应的采样对齐。例如：

- 图像尺寸：123 x 321
- 输入色度子采样：MCJPEG_CSS_410
- 色度采样因子：4 x 2
- 鉴于上述情况，编码器库要求用户提供：
  - Y 平面大小： 123 x 321

- Cb 和 Cr 平面大小：31 x 161

3.1.3.2 mcjpegEncodeImage

此函数的输入（即如何在 source 参数中提供数据）由 input_format 参数决定。对于交错格式（以 I 结尾），只使用第一个通道。对于非交错格式，将使用输入格式中的所有通道。

例如，如果用户连续交错存储了大小为 W x H 的 RGB 图像，且指向该图像的指针是 pImage，则 source 应为：

```
source.channel[0] = pImage
source.pitch[0] = W*3
```

当相同的图像以平面格式存储，并且图像平面指针连续存储在数组 pImage[3]中时，source 应为：

```
source.channel[0] = pImage[0]
source.channel[1] = pImage[1]
source.channel[2] = pImage[2]
```

source 参数中每个通道的 pitch 值应根据数据布局进行相应设置。

3.1.4 获取压缩流

对于任何输入数据和参数，通常无法准确预测最终 JPEG 流的最终压缩数据大小。编码时，mcJPEG 库计算最终数据流的大小，在编码器状态下分配临时缓冲区，并将压缩数据保存在编码状态的缓冲区中。为了获得最终的压缩 JPEG 流，用户应提供足够大的内存缓冲区来存储此压缩数据。执行此操作有两个选择：

- 使用给定参数和图像尺寸下压缩 JPEG 流大小的上限：
  a. 使用 mcjpegEncodeGetBufferSize()函数获取压缩后的 JPEG 流最大可能的大小。
  b. 分配对应大小的内存缓冲区。
  c. 使用一个编码函数对图像进行编码。
  d. 使用 mcjpegEncodeRetrieveBitstream()和分配的缓冲区，在成功编码后从编码器状态中获取压缩的 JPEG 流。
- 等待编码完成，获取所需缓冲区的确切大小，如下所示：
  a. 使用一个编码函数对图像进行编码。
  b. 使用 mcjpegEncodeRetrieveBitstream()函数获取压缩 JPEG 流的大小，以字节为单位。
  c. 分配内存缓冲区，最少为获取到的大小。
  d. 使用 mcjpegEncodeRetrieveBitstream()函数用压缩的 JPEG 流填充缓冲区。

3.1.5 mcJPEG 编码示例

```
mcjpegHandle_t handle;
mcjpegEncoderState_t enc_state;
mcjpegEncoderParams_t enc_params;
mcStream_t stream;

//初始化 mcjpeg 结构体
mcjpegCreateSimple(&handle);
mcjpegEncoderStateCreate(handle, &enc_state, stream);
mcjpegEncoderParamsCreate(handle, &enc_params, stream);

mcjpegImage_t image;
//用图像数据填充图像，比如，640x480 的 RGB 图像

//压缩图像
mcjpegEncodeImage(handle, enc_state, enc_params,
    &image, MCJPEG_INPUT_RGB, 640, 480, stream);

//获取压缩的流大小
size_t length;
mcjpegEncodeRetrieveBitstream(handle, enc_state, NULL, &length, stream);
//获取流本身
mcStreamSynchronize(stream);
std::vector<char> jpeg(length);
mcjpegEncodeRetrieveBitstream(handle, enc_state, jpeg.data(), &length, 0);

//将流写入文件
mcStreamSynchronize(stream);
std::ofstream output_file("test.jpg", std::ios::out | std::ios::binary);
output_file.write(jpeg.data(), length);
output_file.close();
```

3.2 mcJPEG 编码类型声明

3.2.1 mcjpegInputFormat_t

```
typedef enum {
MCJPEG_INPUT_RGB        = 3,
MCJPEG_INPUT_BGR        = 4,
MCJPEG_INPUT_RGBI       = 5,
MCJPEG_INPUT_BGRI       = 6
} mcjpegInputFormat_t;
```

`mcjpegInputFormat_t` 枚举用于选择输入图像的像素格式。

表 3-1 mcjpegInputFormat_t 枚举项

成员	描述
MCJPEG_INPUT_RGB	输入图像为 RGB 色彩模型，像素格式为 RGB
MCJPEG_INPUT_BGR	输入图像为 RGB 色彩模型，像素格式为 BGR
MCJPEG_INPUT_RGBI	输入图像为 RGB 色彩模型，像素格式为交错 RGB
MCJPEG_INPUT_BGRI	输入图像为 RGB 色彩模型，像素格式为交错 BGR

3.2.2 mcjpegEncoderState_t

`mcjpegEncoderState_t` 结构体存储用于压缩的中间缓冲区和变量。

3.2.3 mcjpegEncoderParams_t

`mcjpegEncoderParams_t` 结构体存储 JPEG 编码参数。

3.3 mcJPEG 编码 API

3.3.1 mcjpegEncoderStateCreate()

创建编码状态，用于存储压缩中使用的中间缓冲区。

签名

```
mcjpegStatus_t mcjpegEncoderStateCreate(
mcjpegHandle_t handle,
mcjpegEncoderState_t *encoder_state,
mcStream_t stream);
```

参数

参数	输入/输出	内存	描述
handle	输入	主机	库句柄
encoder_state	输出	主机	此指针指向编码状态结构体
stream	输入	主机	MXMACA 流，包含所有必需的设备操作

3.3.2 mcjpegEncoderStateDestroy()

销毁编码状态。

签名

```
mcjpegStatus_t mcjpegEncoderStateDestroy(    mcjpegEncoderState_t
encoder_state);
```

参数

参数	输入/输出	内存	描述
encoder_state	输入/输出	主机	要被销毁的编码状态结构体

3.3.3 mcjpegEncoderParamsCreate()

创建包含压缩参数的结构体。

签名

```
mcjpegStatus_t mcjpegEncoderParamsCreate(
mcjpegHandle_t handle,
mcjpegEncoderParams_t *encoder_params,
mcStream_t stream);
```

参数

参数	输入/输出	内存	描述
handle	输入	主机	库句柄
encoder_params	输出	主机	此指针指向创建的压缩参数结构体
stream	输入	主机	MXMACA 流，包含所有必需的设备操作

3.3.4 mcjpegEncoderParamsDestroy()

销毁编码参数结构体。

签名

```
mcjpegEncoderParamsDestroy(    mcjpegEncoderParams_t encoder_params);
```

参数

参数	输入/输出	内存	描述
encoder_params	输入/输出	主机	要被销毁的编码器参数结构体

3.3.5 mcjpegEncoderParamsSetEncoding()

在编码参数结构体中设置编码类型参数。

签名

```
mcjpegStatus_t mcjpegEncoderParamsSetEncoding(
```

```
mcjpegEncoderParams_t encoder_params,
mcjpegJpegEncoding_t etype,
mcStream_t stream);
```

参数

参数	输入/输出	内存	描述
encoder_params	输入/输出	主机	编码参数结构体句柄
etype	输入	主机	编码类型选择。 默认为 baseline，当前仅支持 baseline
stream	输入	主机	MXMACA 流，包含所有必需的设备操作

3.3.6 mcjpegEncoderParamsSetQuality()

在编码参数结构体中设置编码质量参数。

签名

```
mcjpegStatus_t mcjpegEncoderParamsSetQuality(
mcjpegEncoderParams_t encoder_params,
const int quality,
mcStream_t stream);
```

参数

参数	输入/输出	内存	描述
encoder_params	输入/输出	主机	编码参数结构体句柄
quality	输入	主机	介于 1 到 100 之间、表示质量的整数值，100 表示最高质量。默认值为 70
stream	输入	主机	MXMACA 流，包含所有必需的设备操作

3.3.7 mcjpegEncoderParamsSetSamplingFactors()

设置用于 JPEG 压缩的色度子采样。

签名

```
mcjpegStatus_t mcjpegEncoderParamsSetSamplingFactors(
mcjpegEncoderParams_t encoder_params,
const mcjpegChromaSubsampling_t chroma_subsampling,
mcStream_t stream);
```

参数

参数	输入/输出	内存	描述
encoder_params	输入/输出	主机	编码参数结构体句柄

参数	输入/输出	内存	描述
chroma_subsampling	输入	主机	用于 JPEG 压缩的色度子采样。如果输入是 YUV 色彩模型，且 `chroma_subsampling` 与源图像的采样因子不同，则 mcJPEG 库会将源图像转换为 `chroma_subsampling` 采样类型。默认值为 4:2:0，当前仅支持 4:2:0
stream	输入	主机	MXMACA 流，包含所有必需的设备操作

3.3.8 mcjpegEncodeGetBufferSize()

为给定的输入参数返回存储压缩 JPEG 流所需的最大可能缓冲区大小。

签名

```
mcjpegStatus_t mcjpegEncodeGetBufferSize(
mcjpegHandle_t handle,
const mcjpegEncoderParams_t encoder_params,
int image_width,
int image_height,
size_t *max_stream_length);
```

参数

参数	输入/输出	内存	描述
handle	输入	主机	库句柄
encoder_params	输入/输出	主机	编码参数结构体句柄
image_width	输入	主机	输入图像宽度
image_height	输入	主机	输入图像高度
stream	输入	主机	MXMACA 流，包含所有必需的设备操作

3.3.9 mcjpegEncodeYUV()

使用提供的参数将 YUV 图像压缩到 JPEG 流，并将其存储在状态结构体中。

签名

```
mcjpegStatus_t mcjpegEncodeYUV(
mcjpegHandle_t handle,
mcjpegEncoderState_t encoder_state,
const mcjpegEncoderParams_t encoder_params,
const mcjpegImage_t *source,
mcjpegChromaSubsampling_t chroma_subsampling,
int image_width,
int image_height,
mcStream_t stream);
```

参数

参数	输入/输出	内存	描述
handle	输入	主机	库句柄
encoder_state	输入/输出	主机	内部结构体，用于保存压缩所需的临时缓冲区，并存储最终压缩的 JPEG 流
encoder_params	输入	主机	编码参数结构体句柄
source	输入	主机	此指针指向 `mcjpegImage_t` 结构体
chroma_subsampling	输入	主机	输入数据的色度子采样
image_width	输入	主机	输入图像宽度
image_height	输入	主机	输入图像高度
stream	输入	主机	MXMACA 流，包含所有必需的设备操作

3.3.10 mcjpegEncodeImage()

使用提供的参数将 RGB 图像压缩到 JPEG 流，并将其存储在状态结构体中。

签名

```
mcjpegStatus_t mcjpegEncodeImage(
mcjpegHandle_t handle,
mcjpegEncoderState_t encoder_state,
const mcjpegEncoderParams_t encoder_params,
const mcjpegImage_t *source,
mcjpegInputFormat_t input_format,
int image_width,
int image_height,
mcStream_t stream);
```

参数

参数	输入/输出	内存	描述
handle	输入	主机	库句柄
encoder_state	输入/输出	主机	内部结构体，用于保存压缩所需的临时缓冲区，以及存储最终压缩的 JPEG 流
encoder_params	输入	主机	编码参数结构体句柄
source	输入	主机	此指针指向 `mcjpegImage_t` 结构体
input_format	输入	主机	描述输入数据的 `mcjpegInputFormat_t` 类型的值
image_width	输入	主机	输入图像宽度
image_height	输入	主机	输入图像高度

参数	输入/输出	内存	描述
stream	输入	主机	MXMACA 流，包含所有必需的设备操作

3.3.11 mcjpegEncodeRetrieveBitstream()

从先前在一个编码函数中使用过的编码状态中获取压缩流。

- 如果 `data` 参数为 NULL，则在 `length` 参数中返回压缩流的大小。
- 如果 `data` 不为空，则提供的 `length` 参数应包含 `data` 缓冲区大小。
- 如果提供的 `length` 小于压缩流的大小，则返回错误。否则，压缩流将存储在 `data` 缓冲区中，且实际的压缩缓冲区大小将存储在 `length` 参数中。

签名

```
mcjpegStatus_t mcjpegEncodeRetrieveBitstream(
mcjpegHandle_t handle,
mcjpegEncoderState_t encoder_state,
unsigned char *data,
size_t *length,
mcStream_t stream);
```

参数

参数	输入/输出	内存	描述
handle	输入	主机	库句柄
encoder_state	输入/输出	主机	先前在一个编码函数中使用过的 `encoder_state`
data	输入/输出	主机	此指针指向主机内存中用于存储压缩流的缓冲区。可以为 NULL
length	输入/输出	主机	此指针指向输入缓冲区大小。返回时，mcJPEG 库将在此参数中存储实际的压缩流大小
stream	输入	主机	MXMACA 流，包含所有必需的设备操作

3.3.12 mcjpegEncodeRetrieveBitstreamDevice()

从先前在一个编码函数中使用过的编码状态中获取压缩流。

- `data` 参数应指向设备内存。
- 如果 `data` 参数为 NULL，则将在 `length` 参数中返回压缩流的大小。
- 如果 `data` 不为空，则提供的 `length` 参数应包含 `data` 缓冲区大小。
- 如果提供的 `length` 小于压缩流的大小，则返回错误。否则，压缩的流将存储在 `data` 缓冲区中，而实际的压缩缓冲区大小将存储在 `length` 参数中。

签名

```
mcjpegStatus_t mcjpegEncodeRetrieveBitstreamDevice(
mcjpegHandle_t handle,
mcjpegEncoderState_t encoder_state,
unsigned char *data,
size_t *length,
mcStream_t stream);
```

参数

参数	输入/输出	内存	描述
handle	输入	主机	库句柄
encoder_state	输入/输出	主机	先前在一个编码函数中使用过的 `encoder_state`
data	输入/输出	设备	此指针指向设备内存中用于存储压缩流的缓冲区。可以为 NULL
length	输入/输出	主机	此指针指向输入缓冲区大小。返回时，mcJPEG 库将在此参数中存储实际的压缩流大小
stream	输入	主机	MXMACA 流，包含所有必需的设备操作

声明