# [CUDA 纹理内存](https://www.cnblogs.com/liangliangdetianxia/p/4198838.html)

## **1、概述**

纹理存储器中的数据以一维、二维或者三维数组的形式存储在显存中，可以通过缓存加速访问，并且可以声明大小比常数存储器要大的多。

在kernel中访问纹理存储器的操作称为**纹理拾取(texture fetching)**。将显存中的数据与纹理参照系关联的操作，称为将数据与**纹理绑定(texture binding)**.

显存中可以绑定到纹理的数据有两种，分别是普通的线性存储器和cuda数组。

**注**：线性存储器只能与一维或二维纹理绑定,采用整型纹理拾取坐标，坐标值与数据在存储器中的位置相同；

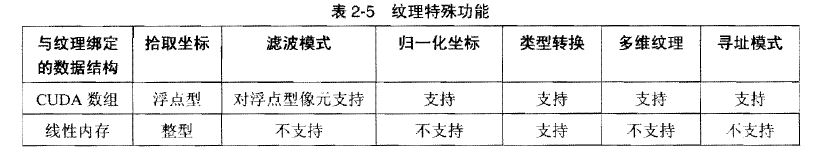
CUDA数组可以与一维、二维、三维纹理绑定，纹理拾取坐标为归一化或者非归一化的浮点型，并且支持许多特殊功能。

## **2、纹理缓存**：

（1）、纹理缓存中的数据可以被重复利用

（2）、纹理缓存一次预取拾取坐标对应位置附近的几个象元，可以实现滤波模式。

## **3、纹理存储器的特殊功能**



## **4、纹理存储器的使用**

使用纹理存储器时，首先要在主机端声明要绑定到纹理的线性存储器或CUDA数组。

(1)声明纹理参考系

texture<Type, Dim, ReadMode> texRef;

//Type指定数据类型，特别注意：不支持3元组

//Dim指定纹理参考系的维度，默认为1

//ReadMode可以是cudaReadModelNormalizedFloat或cudaReadModelElementType (默认)

注：纹理参照系必须定义在所有函数体外

(2) 声明CUDA数组，分配空间

CUDA数组可以通过cudaMalloc3DArray()或者cudaMallocArray()函数分配。前者可以分配1D、2D、3D的数组，后者一般用于分配2D的CUDA数组。使用完毕，要用cudaFreeArray()函数释放显存。

//1数组

cudaMalloc((void\*\*)&dev\_A, data\_size);

cudaMemcpy(dev\_A, host\_A, data\_size, cudaMemcpyHostToDevice);

cudaFree(dev\_A);

//2维数组

cudaChannelFormatDesc channelDesc = cudaCreateChannelDesc<float>()

cudaArray \*cuArray;

cudaMallocArray(&cuArray, &channelDesc, 64, 32); //64x32

cudaMemcpyToArray(cuArray, 0, 0, h\_data, sizeof(float)\*width\*height, cudaMemcpyHostToDevice);

cudaFreeArray(cuArray);

//3维数组 64x32x16

cudaChannelFormatDesc channelDesc = cudaCreateChannelDesc<uchar>();

cudaArray \*d\_volumeArray;

cudaMalloc3DArray(&d\_volumeArray, &channelDesc, volumSize);

cudaMemcpy3DParms copyParams = {0};

copyParams.srcPtr=make\_cudaPitchedPtr((void\*)h\_volume, volumeSize.width\*sizeof(uchar), volumeSize.width, volumeSize.height);

copyParams.dstArray = d\_volumeArray;

copyParams.extent = volumeSize;

copyParams.kind = cudaMemcpyHostToDevice;

cudaMemcpy3D(&copyParams);

tex.normalized = true;

tex.filterMode = cudaFilterModeLinear;

tex.addressMode[0] = cudaAddressModeWrap;

tex.addressMode[1] = cudaAddressModeWrap;

tex.addressMode[2] = cudaAddressModeWrap;

（3）设置运行时纹理参照系属性

struct textureReference{

int normalized;

enum cudaTextureFilterMode filterMode;

enum cudaTextureAddressMode addressMode[3];

struct cudaChannelFormatDesc channelDesc;

}

normalized设置是否对纹理坐标归一化

filterMode用于设置纹理的滤波模式

addressMode说明了寻址方式

(4)纹理绑定

通过cudaBindTexture() 或 cudaBindTextureToArray()将数据与纹理绑定。

通过cudaUnbindTexture()用于解除纹理参照系的绑定

注：与纹理绑定的数据的类型必须与声明纹理参照系时的参数匹配

(I).cudaBindTexture() //将1维线性内存绑定到1维纹理

cudaError\_t cudaBindTexture(

size\_t \* offset,

const struct textureReference \* texref,

const void \* devPtr,

const struct cudaChannelFormatDesc \* desc,

size\_t size = UINT\_MAX

)

cudaMalloc((void\*\*)&data.dev\_inSrc, imageSize);

cudaBindTexture(NULL, tex, data.dev\_inSrc, imageSize);

(II).cudaBindTexture2D //将1维线性内存绑定到2维纹理

cudaError\_t cudaBindTexture2D(

size\_t \* offset,

const struct textureReference \* texref,

const void \* devPtr,

const struct cudaChannelFormatDesc \* desc,

size\_t width,

size\_t height,

size\_t pitch

)

cudaMalloc((void\*\*)&data.dev\_inSrc, imageSize);

cudaChannelFormatDesc desc = cudaCreateChannelDesc<float>();

cudaBindTexture2D(NULL, tex, data.dev\_inSrc, desc, DIM, DIM, sizeof(float)\*DIM);

(III). cudaBindTextureToArray() //将cuda数组绑定到纹理

cudaError\_t cudaBindTextureToArray (

const struct textureReference \* texref,

const struct cudaArray \* array,

const struct cudaChannelFormatDesc \* desc

)

void initCudaTexture(const uchar \*h\_volume, cudaExtent volumeSize)

{

cudaChannelFormatDesc channelDesc = cudaCreateChannelDesc<uchar>();

cutilSafeCall(cudaMalloc3DArray(&d\_volumeArray, &channelDesc, volumeSize));

cudaMemcpy3DParms copyParams = {0};

copyParams.srcPtr = make\_cudaPitchedPtr((void\*)h\_volume, volumeSize.width\*sizeof(uchar), volumeSize.width, volumeSize.height);

copyParams.dstArray = d\_volumeArray;

copyParams.extent = volumeSize;

copyParams.kind = cudaMemcpyHostToDevice;

cutilSafeCall(cudaMemcpy3D(&copyParams));

tex.normalized = true;

tex.filterMode = cudaFilterModeLinear;

tex.addressMode[0] = cudaAddressModeWrap;

tex.addressMode[1] = cudaAddressModeWrap;

tex.addressMode[2] = cudaAddressModeWrap;

cutilSafeCall(cudaBindTextureToArray(tex, d\_volumeArray, channelDesc));

}

(5)纹理拾取

对于线性存储器绑定的纹理，使用tex1Dfetch()访问，采用的纹理坐标是整型。由cudaMallocPitch() 或者 cudaMalloc3D()分配的线性空间实际上仍然是经过填充、对齐的一维线性空间，因此也用tex1Dfetch()

对与一维、二维、三维cuda数组绑定的纹理，分别使用tex1D(), tex2D() 和 tex3D()函数访问，并且使用浮点型纹理坐标。