



Denglin Hamming™ V2

Texture Feature Guide

DL-DG/SW-042A-04

2024-07-24

Copyright©苏州登临科技有限公司，2019 - 2025，版权所有。

未经苏州登临科技有限公司事先书面同意，不得以任何形式或方式复制或传播本文件的任何部分。

商标和许可



和其它苏州登临科技有限公司的其它登临科技的图标为苏州登临科技有限公司的商标。本手册中提及的所有其他商标均为其各自所有者的财产。

通知

所购买的产品、服务和特性由苏州登临科技有限公司与客户签订的合同规定。本文件中描述的所有或部分产品、服务和特性可能不在采购范围或使用范围内。除非合同中另有规定，本文件中的所有声明、信息和建议均按“原样”提供，无任何明示或暗示的保证或陈述。

本手册中的信息如有更改，恕不另行通知。本文件在编制过程中已尽一切努力确保内容的准确性，本文件中的所有声明、信息和建议不构成任何明示或暗示的保证。

苏州登临科技有限公司

苏州工业园区扬富路11号南岸新地一期商务楼5号1101室，江苏，中国

<http://www.denglin.ai>

email : support@denglin.ai

变更历史

版本	变更描述
04	增加tex3D支持 float。
03	适配CUDA 11。
02	tex2D 效率加速，优化内容。适用于 Denglin Texture V1.5。
01	初始版本。

内容目录

1 概述

- 1.1 CUDA Texture
- 1.2 Denglin Texture

2 Texture 指令

- 2.1 指令支持说明
- 2.2 指令参数支持说明
- 2.3 指令加速支持说明
- 2.4 Texture 使用示例

3 Texture 应用模式

- 3.1 Texture Object 模式
- 3.2 Texture Reference 模式

附录 A

- A.1 不支持的 Texture 指令

1 概述

1.1 CUDA Texture

CUDA Texture 的详细介绍，请参考如下 NVIDIA 官方文档：

- [《CUDA C++ Programming Guide》 - Texture and Surface Memory](#)
- [《CUDA C++ Best Practices Guide》 - Texture Memory](#)

1.2 Denglin Texture

- Denglin Texture 主要应用于深度学习推理的预处理与后处理模块。
- Denglin Texture 当前版本（V1.5）只支持 CUDA Texture 部分特性，具体支持情况如下表。

Texture 特性项	支持说明
Texture 指令	1) 1D 类指令，只支持 tex1Dfetch，其他指令都不支持。 2) 2D 类指令，只支持 tex2D 与 tex2DLayerd，其他指令不支持。 3) 3D 类指令，只支持 tex3D，其他指令不支持。 4) Cubemap 类指令，都不支持。 详细内容参见章节 2.1 指令支持说明
Texture Sample 参数	1) tex2D 指令，支持常用的 sample 参数。 2) 1D 与 3D 指令，只支持特定 sample 参数。 详细内容参见章节 2.2 指令参数支持说明
Resource Type	1) tex2D 指令，支持常用的 ResourceType。 2) 1D指令，只支持 uchar Type。 3) 3D指令，只支持 uchar和float Type。 详细内容参见章节 2.2 指令参数支持说明
Texture Mode	1) 只支持 Texture Object 模式，不支持 Reference 模式。 2) 只支持常用 TextureObject 传递到 Kernel 的模式。 详细内容参见章节 3 Texture 应用模式
cudaCreateTextureObject API	不支持 ResourceView Create 模式（即：不支持参数 <code>cudaResourceViewDesc</code> ）。

2 Texture 指令

2.1 指令支持说明

Denglin Texture 当前版本 (V1.5) 支持的 Texture 指令，详细说明如下表所示。

Denglin Texture 当前版本 (V1.5) 不支持的 Texture 指令，请参考附录A.1。

指令类别	Texture 指令名称	Texture 指令 API	Texture Mode	是否支持	支持的数据类型
1D	tex1Dfetch	tex1Dfetch(T *ptr, cudaTextureObject_t obj, int x) T tex1Dfetch(cudaTextureObject_t texObject, int x)	Texture Object	YES	uchar
2D	tex2D	tex2D(T *ptr, cudaTextureObject_t texObj, float x, float y) T tex2D(cudaTextureObject_t texObj, float x, float y)	Texture Object	YES	uchar ushort float uchar4 ushort4 float4
2D	tex2DLayered	tex2DLayered(T *ptr,cudaTextureObject_t texObj,float x, float y, int layer) T tex2DLayered(cudaTextureObject_t texObj,float x, float y, int layer)	Texture Object	YES	uchar
3D	tex3D	tex3D(T *ptr, cudaTextureObject_t texObj, float x, float y, float z) T tex3D(cudaTextureObject_t texObj, float x, float y, float z)	Texture Object	YES	uchar float

2.2 指令参数支持说明

Denglin Texture 当前版本 (V1.5) 支持的指令参数，详细说明如下表所示。

Tex 指令	参数类别	参数名称	支持的参数项
tex2D	Image Description	DataType	支持：uchar, uchar4, ushort, ushort4, float, float4 不支持：上述类型外的其他 datatype (如 char, uchar2, short, ushort2, int 等)
		ResourceType	支持 cudaArray , Pitch2D , Linear
	Sample Param	ReadMode	支持 CUDA 的所有类型 (Element 与 Normal)
		FilterMode	支持 CUDA 的所有类型 (Nearest 与 Linear)

		NormalizeCoords	支持：FALSE 不支持：TRUE
		AddressMode	支持：CLAMP 不支持：TRANSPRANT，MIRROR，CONSTANT
		Adv Sample Param(sRGB, maxAnisotropy, mipMap)	支持 CUDA 默认配置，不支持其他配置方式
tex1Dfetch tex2DLayered tex3D	Image Description	DataType	3D支持uchar和float，1D与2D只支持uchar，不支持其他类型
		ResourceType	1D 只支持 Linear，2D 与 3D 只支持 cudaArray
	Sample Param	ReadMode	支持 Element，不支持 Normal
		FilterMode	支持Nearest，不支持Linear。3D在DataType为float时支持Nearest和Linear
		NormalizeCoords	支持 FALSE，不支持 TRUE
		AddressMode	支持 CLAMP，不支持其他类型。3D在DataType为float时支持CLAMP和BORDER
		Adv Sample Param(sRGB, maxAnisotropy, mipMap)	只支持 CUDA 默认配置，不支持其他配置方式

2.3 指令加速支持说明

- Texture 指令加速版本：开启编译宏 " __DLGPU_TEXTURE_BIND_MODE__ "，Texture 指令会被编译成 img_ldg (surface 指令) 硬件指令进行加速。
- 编译命令示例如下：

```
#启用编译宏 __DLGPU_TEXTURE_BIND_MODE__  
dlcc -D__DLGPU_TEXTURE_BIND_MODE__ --cuda-gpu-arch=dlgput64 -x cuda texture_test.cu
```

- Denglin Texture 当前版本（V1.5），只对常用的 Texture 指令进行加速，加速说明如下表：

序号	Texture 指令	加速支持说明
1	Tex1Dfetch	不支持
2	Tex2D	只支持 Datatype size 小于等于 4Byte 的指令加速，包括 uchar, uchar, ushort, float 不支持 Datatype size 大于 4Byte 的指令加速, 包括 ushort4，float4

序号	Texture 指令	加速支持说明
3	tex2DLayered	不支持
4	Tex3D	不支持

2.4 Texture 使用示例

使用 Texture 实现 resize and make border 功能，示例代码如下：

```
// Simple image resize and make border kernel
global void resizeMakeBorderKernel (cudaTextureObject_t texobj, uint8_t* dst_gray,
                                     int dst_pitch, float x_scale, float y_scale, int pad)
{
    unsigned int x = blockIdx.x * blockDim.x + threadIdx.x;
    unsigned int y = blockIdx.y * blockDim.y + threadIdx.y;
    float src_x = x * x_scale - pad;
    float src_y = y * y_scale - pad;
    dst_gray[y * dst_pitch + x] = tex2D<uint8_t>(texobj, src_x, src_y);
}

int main()
{
    int src_width  = 320;
    int src_height = 240;
    int dst_width  = 232;
    int dst_height = 192;
    int dst_image_pad = 16;
    float resize_scale_x = src_width / (float)(dst_width - dst_image_pad * 2);
    float resize_scale_y = src_height / (float)(dst_height - dst_image_pad * 2);

    // Allocate and set some host data
    uint8_t* h_src_image = (uint8_t*)std::malloc(src_width * src_height *
    sizeof(uint8_t));
    for (int i = 0; i < src_width * src_height; ++i)
        h_src_image[i] = (uint8_t)(i);

    // Allocate CUDA array in device memory
    cudaChannelFormatDesc channelDesc=cudaCreateChannelDesc(8, 0, 0, 0,
    cudaChannelFormatKindUnsigned);
    cudaArray* cuarr_inping;
    cudaMallocArray(&cuarr_inping, &channelDesc, src_width * sizeof(uint8_t),
    src_height);
    // Copy data located at address h_src_image in host memory to device memory
    cudaMemcpyToArray(cuarr_inping, 0, 0, h_src_image,
        src_width * src_height * sizeof(uint8_t), cudaMemcpyHostToDevice);

    // Specify texture
    cudaResourceDesc texRes;
    memset(&texRes, 0, sizeof(cudaResourceDesc));
```



```

texRes.resType = cudaResourceTypeArray;
texRes.res.array.array = cuarr_inimg;

// Specify texture object parameters cudaTextureDesc texDescr;
cudaTextureDesc          texDescr;
memset(&texDescr, 0, sizeof(cudaTextureDesc));
texDescr.normalizedCoords = false;
texDescr.filterMode = cudaFilterModePoint;
texDescr.addressMode[0] = cudaAddressModeClamp;
texDescr.addressMode[1] = cudaAddressModeClamp;
texDescr.readMode = cudaReadModeElementType;

// Create texture object
cudaTextureObject_t texobj_host;
cudaCreateTextureObject(&texobj_host, &texRes, &texDescr, nullptr);

// Allocate result of resized image in device memory size t dst_pitch; uint8_t*
d_outbuf;
size_t dst_pitch;
uint8_t* d_outbuf;

cudaMallocPitch(&d_outbuf, &dst_pitch, dst_width, dst_height);

// Invoke kernel
dim3 blocksize(8, 8);
dim3 gridsize(dst_width / blocksize.x, dst_height / blocksize.y);
resizeMakeBorderKernel <<<gridsize, blocksize >>> (texobj_host, d_outbuf, dst_pitch,
                                                    resize_scale_x, resize_scale_y, dst_image_pad);

//Allocate output host buffer, Copy data from device back to host
uint8_t* h_output = (uint8_t*)std::malloc(dst_width * dst_height * sizeof(uint8_t));
cudaMemcpy2D(h_output, dst_width, d_outbuf,
             dst_pitch, dst_width, dst_height, cudaMemcpyDeviceToHost);

// Destroy texture object
cudaFreeArray(cuarr_inimg);
cudaFree(d_outbuf);

free(h_output);
free(h_src_image);

return 1;
}

```

3 Texture 应用模式

Denglin Texture 当前版本 (V1.5) 只支持 Texture Obeject 模式，不支持 Texture Reference 模式。

3.1 Texture Object 模式

Denglin Texture 当前版本 (V1.5)，只支持部分 Object 封装方式，如下表：

传入到 Kernel 的方式	Texture Object 封装方式	是否支持
通过 argument 传入到 Kernel	texobj 类型变量	Yes
	struct (texobj 作为成员变量)	Yes
	class object (texobj 作为成员变量)	No
	数组 (包括 texobj 数组，texobj struct 数组，texobj class 数组)	No
通过 device global 传入到 Kernel	texobj 类型变量	Yes
	struct (texobj 作为成员变量)	No
	class object (texobj 作为成员变量)	No
	数组 (包括 texobj 数组，texobj struct 数组，texobj class 数组)	No

3.2 Texture Reference 模式

- Denglin Texture 当前版本 (V1.5)，不支持 Texture Reference 模式，但可通过 device global texture object 来代替 Reference 模式。
- 从 Host 将 host texture object 拷贝到 device global texture object 功能，当前版本 (V1.5) 只支持 cudaMemcpyToSymbol 的方式，不支持 cudaGetSymbolAddress+cudaMemcpy 的方式。
- 使用 device global texture object 实现 resize and make border 功能，示例代码如下：

```
//Declare global textureObject on device
__device__ cudaTextureObject_t g_texobj_dev;

// Simple image resize and make border kernel
__global__ void resizeMakeBorderKernel(uint8_t* dst_gray, int dst_pitch,
                                       float x_scale, float y_scale, int pad)
{
    unsigned int x = blockIdx.x * blockDim.x + threadIdx.x;
    unsigned int y = blockIdx.y * blockDim.y + threadIdx.y;
    float src_x = x * x_scale - pad;
    float src_y = y * y_scale - pad;
    dst_gray[y * dst_pitch + x] = tex2D<uint8_t>(g_texobj_dev, src_x, src_y);
}
```

```

}

int main()
{
    int src_width  = 320;
    int src_height = 240;
    int dst_width  = 232;
    int dst_height = 192;
    int dst_image_pad = 16;
    float resize_scale_x = src_width / (float)(dst_width - dst_image_pad * 2);
    float resize_scale_y = src_height / (float)(dst_height - dst_image_pad * 2);

    // Allocate and set some host data
    uint8_t* h_src_image = (uint8_t*)std::malloc(src_width * src_height *
sizeof(uint8_t));
    for (int i = 0; i < src_width * src_height; ++i)
        h_src_image[i] = (uint8_t)(i);

    // Allocate CUDA array in device memory
    cudaChannelFormatDesc channelDesc=cudaCreateChannelDesc(8, 0, 0, 0,
cudaChannelFormatKindUnsigned);
    cudaArray* cuarr_inping;
    cudaMallocArray(&cuarr_inping, &channelDesc, src_width * sizeof(uint8_t),
src_height);

    // Copy data located at address h_src_image in host memory to device memory
    cudaMemcpyToArray(cuarr_inping, 0, 0, h_src_image,
src_width * src_height * sizeof(uint8_t),
cudaMemcpyHostToDevice);

    // Specify texture
    cudaResourceDesc texRes;
    memset(&texRes, 0, sizeof(cudaResourceDesc));

    texRes.resType          = cudaResourceTypeArray;
    texRes.res.array.array = cuarr_inping;

    // Specify texture object parameters cudaTextureDesc texDescr;
    cudaTextureDesc texDescr;
    memset(&texDescr, 0, sizeof(cudaTextureDesc));
    texDescr.normalizedCoords = false;
    texDescr.filterMode       = cudaFilterModePoint;
    texDescr.addressMode[0]   = cudaAddressModeClamp;
    texDescr.addressMode[1]   = cudaAddressModeClamp;
    texDescr.readMode         = cudaReadModeElementType;

    // Create texture object
    cudaTextureObject_t texobj_host;
    cudaCreateTextureObject(&texobj_host, &texRes, &texDescr, nullptr);

    // Allocate result of resized image in device memory size t dst_pitch; uint8_t*
d_outbuf;
    size_t dst_pitch;
    uint8_t* d_outbuf;

```

```

        cudaMallocPitch(&d_outbuf, &dst_pitch, dst_width, dst_height);

        //Copy texobj from host to device symbol
        cudaMemcpyToSymbol(g_texobj_dev, (const void*)&texobj_host,
sizeof(cudaTextureObject_t));

        // Invoke kernel
        dim3 blocksize(8, 8);
        dim3 gridsize(dst_width / blocksize.x, dst_height / blocksize.y);

        resizeMakeBorderKernel <<<gridsize, blocksize >>> (d_outbuf, dst_pitch,
                                                                resize_scale_x, resize_scale_y,
dst_image_pad);

        //Allocate output host buffer, Copy data from device back to host
        uint8_t* h_output = (uint8_t*)std::malloc(dst_width * dst_height *
sizeof(uint8_t));
        cudaMemcpy2D(h_output, dst_width, d_outbuf, dst_pitch,
                                                                dst_width, dst_height, cudaMemcpyDeviceToHost);

        // Destroy texture object
        cudaFreeArray(cuarr_inpimg);
        cudaFree(d_outbuf);

        free(h_output);
        free(h_src_image);

        return 1;
    }

```

附录 A

A.1 不支持的 Texture 指令

Denglin Texture 当前版本 (V1.5) 不支持的 Texture 指令如下表：

指令类别	Texture 指令名称	Texture 指令 API	Texture Mode	是否支持	支持的数据类型
1D	tex1Dfetch	__nv_tex_rmet_ret< T> tex1Dfetch(texture< T, cudaTextureType1D, cudaReadModeElementType> t, int x);	Texture Reference	NO	-
		__nv_tex_rmnf_ret< T> tex1Dfetch(texture< T, cudaTextureType1D, cudaReadModeNormalizedFloat> t, int x)			
	tex1D	__nv_tex_rmet_ret< T> tex1D(texture< T, cudaTextureType1D, cudaReadModeElementType> t, float x);	Texture Reference	NO	-
		__nv_tex_rmnf_ret< T> tex1D(texture< T, cudaTextureType1D, cudaReadModeNormalizedFloat> t, float x)			
		tex1D(T *ptr, cudaTextureObject_t texObj, float x);	Texture Object	NO	-
		T tex1D(cudaTextureObject_t texObj, float x)			
	tex1DLod	__nv_tex_rmet_ret< T> tex1DLod(texture< T, cudaTextureType1D, cudaReadModeElementType> t, float x,float level);	Texture Reference	NO	-
		__nv_tex_rmnf_ret< T> tex1DLod(texture< T, cudaTextureType1D, cudaReadModeNormalizedFloat> t, float x,float level)			
		tex1DLod(T *ptr, cudaTextureObject_t texObj, float x, float level);	Texture Object	NO	-
		T tex1DLod(cudaTextureObject_t texObj, float x, float level)			

	tex1DGrad	__nv_tex_rmet_ret< T> tex1DGrad(texture< T, cudaTextureType1D, cudaReadModeElementType> t, float x, float dx, float dy); __nv_tex_rmnf_ret< T> tex1DGrad(texture< T, cudaTextureType1D, cudaReadModeNormalizedFloat> t, float x, float dx, float dy)	Texture Reference	NO	-
		tex1DGrad(T *ptr, cudaTextureObject_t texObj, float x, float dx, float dy); T tex1DGrad(cudaTextureObject_t texObj, float x, float dx, float dy)	Texture Object	NO	-
	tex1DLayered	__nv_tex_rmet_ret< T> tex1DLayered(texture< T, cudaTextureType1DLayered, cudaReadModeElementType> t, float x, int layer); __nv_tex_rmnf_ret< T> tex1DLayered(texture< T, cudaTextureType1DLayered, cudaReadModeNormalizedFloat> t, float x, int layer)	Texture Reference	NO	-
		tex1DLayered(T *ptr, cudaTextureObject_t texObj, float x, int layer); T tex1DLayered(cudaTextureObject_t texObj, float x, int layer)	Texture Object	NO	-
	tex1DLayeredLod	__nv_tex_rmet_ret< T> tex1DLayeredLod(texture< T, cudaTextureType1DLayered, cudaReadModeElementType> t, float x, int layer, float level); __nv_tex_rmnf_ret< T> tex1DLayeredLod(texture< T, cudaTextureType1DLayered, cudaReadModeNormalizedFloat> t, float x, int layer, float level)	Texture Reference	NO	-
		tex1DLayeredLod(T *ptr, cudaTextureObject_t texObj, float x, int layer, float level); T tex1DLayeredLod(cudaTextureObject_t texObj, float x, int layer, float level)	Texture Object	NO	-

	tex1DLayeredGrad	__nv_tex_rmet_ret< T> tex1DLayeredGrad(texture< T, cudaTextureType1DLayered, cudaReadModeElementType> t, float x,int layer, float dx, float dy); __nv_tex_rmnf_ret< T> tex1DLayeredGrad(texture< T, cudaTextureType1DLayered, cudaReadModeNormalizedFloat> t, float x,int layer, float dx, float dy)	Texture Reference	NO	-
		tex1DLayeredGrad(T *ptr, cudaTextureObject_t texObj, float x, int layer,float dx, float dy); T tex1DLayeredGrad(cudaTextureObject_t texObj, float x, int layer,float dx, float dy)	Texture Object	NO	-
2D	tex2D	__nv_tex_rmet_ret< T> tex2D(texture< T, cudaTextureType2D, cudaReadModeElementType> t, float x, float y); __nv_tex_rmnf_ret< T> tex2D(texture< T, cudaTextureType2D, cudaReadModeNormalizedFloat> t, float x, float y)	Texture Reference	NO	-
	tex2DLod	__nv_tex_rmet_ret< T> tex2DLod(texture< T, cudaTextureType2D, cudaReadModeElementType> t, float x, float y, float level); __nv_tex_rmnf_ret< T> tex2DLod(texture< T, cudaTextureType2D, cudaReadModeNormalizedFloat> t, float x, float y, float level)	Texture Reference	NO	-
		tex2DLod(T *ptr, cudaTextureObject_t texObj, float x, float y, float level); T tex2DLod(cudaTextureObject_t texObj, float x, float y, float level)	Texture Object	NO	-
	tex2DGrad	__nv_tex_rmet_ret< T> tex2DGrad(texture< T, cudaTextureType2D, cudaReadModeElementType> t, float x, float y, float2 dx, float2 dy); __nv_tex_rmnf_ret< T> tex2DGrad(texture< T, cudaTextureType2D, cudaReadModeNormalizedFloat> t, float x, float y, float2 dx, float2 dy)	Texture Reference	NO	-

	tex2DGrad(T *ptr, cudaTextureObject_t texObj, float x, float y, float2 dx, float2 dy); T tex2DGrad(cudaTextureObject_t texObj, float x, float y, float2 dx, float2 dy)	Texture Object	NO	-
tex2DLayered	__nv_tex_rmet_ret tex2DLayered(texture t, float x, float y, int layer); __nv_tex_rmnf_ret tex2DLayered(texture t, float x, float y, int layer)	Texture Reference	NO	-
tex2DLayeredLod	__nv_tex_rmet_ret< T> tex2DLayered(texture< T, cudaTextureType2DLayered, cudaReadModeElementType> t, float x, float y, int layer); __nv_tex_rmnf_ret< T> tex2DLayered(texture< T, cudaTextureType2DLayered, cudaReadModeNormalizedFloat> t, float x, float y, int layer)	Texture Reference	NO	-
	tex2DLayered(T *ptr, cudaTextureObject_t texObj, float x, float y, int layer); T tex2DLayered(cudaTextureObject_t texObj, float x, float y, int layer)	Texture Object	NO	uchar
tex2DLayeredGrad	__nv_tex_rmet_ret< T> tex2DLayeredLod(texture< T, cudaTextureType2DLayered, cudaReadModeElementType> t, float x, float y, int layer, float level); __nv_tex_rmnf_ret< T> tex2DLayeredLod(texture< T, cudaTextureType2DLayered, cudaReadModeNormalizedFloat> t, float x, float y, int layer, float level)	Texture Reference	NO	-
	tex2DLayeredLod(T *ptr, cudaTextureObject_t texObj, float x, float y, int layer, float level); T tex2DLayeredLod(cudaTextureObject_t texObj, float x, float y, int layer, float level)	Texture Object	NO	-

	tex2Dgather	__nv_tex_rmet_ret< T> tex2DLayeredGrad(texture< T, cudaTextureType2DLayered, cudaReadModeElementType> t, float x, float y, int layer, float dx, float dy); __nv_tex_rmnf_ret< T> tex2DLayeredGrad(texture< T, cudaTextureType2DLayered, cudaReadModeNormalizedFloat> t, float x, float y, int layer, float dx, float dy)	Texture Reference	NO	-
		tex2DLayeredGrad(T *ptr,cudaTextureObject_t texObj, float x, float y, int layer,float dx, float dy); T tex2DLayeredGrad(cudaTextureObject_t texObj, float x, float y, int layer,float dx, float dy)	Texture Object	NO	-
3D	tex3D	__nv_tex_rmet_ret< T> tex3D(texture< T, cudaTextureType3D, cudaReadModeElementType> t, float x, float y, float z); __nv_tex_rmnf_ret< T> tex3D(texture< T, cudaTextureType3D, cudaReadModeNormalizedFloat> t, float x, float y, float z)	Texture Reference	NO	-
	tex3DLod	__nv_tex_rmet_ret< T> tex3DLod(texture< T, cudaTextureType3D, cudaReadModeElementType> t, float x, float y, float z, float level); __nv_tex_rmnf_ret< T> tex3DLod(texture< T, cudaTextureType3D, cudaReadModeNormalizedFloat> t, float x, float y, float z, float level)	Texture Reference	NO	-
		tex3DLod(T *ptr, cudaTextureObject_t texObj, float x, float y, float z, float level); T tex3DLod(cudaTextureObject_t texObj, float x, float y, float z, float level)	Texture Object	NO	-
	tex3DGrad	__nv_tex_rmet_ret< T> tex3DGrad(texture< T, cudaTextureType3D, cudaReadModeElementType> t, float x, float y, float z, float4 dPdx, float4 dPdy); __nv_tex_rmnf_ret< T> tex3DGrad(texture< T, cudaTextureType3D, cudaReadModeNormalizedFloat> t, float x, float y, float z, float4 dPdx, float4 dPdy)	Texture Reference	NO	-

		tex3DGrad(T *ptr,cudaTextureObject_t texObj, float x, float y, float z,float4 dx, float4 dy); T tex3DGrad(cudaTextureObject_t texObj, float x, float y, float z,float4 dx, float4 dy)	Texture Object	NO	-
Cubemap	texCubemap	__nv_tex_rmet_ret< T> texCubemap(texture< T, cudaTextureType3D, cudaReadModeElementType> t, float x, float y, float z); __nv_tex_rmnf_ret< T> texCubemap(texture< T, cudaTextureType3D, cudaReadModeNormalizedFloat> t, float x, float y, float z)	Texture Reference	NO	-
		texCubemap(T *ptr, cudaTextureObject_t texObj, float x, float y, float z); T texCubemap(cudaTextureObject_t texObj, float x, float y, float z)	Texture Object	NO	-
	texCubemapGrad()	__nv_tex_rmet_ret< T> texCubemapGrad(texture< T, cudaTextureType3D, cudaReadModeElementType> t, float x, float, y, float z,float4 dx, float4 dy); __nv_tex_rmnf_ret< T> texCubemapGrad(texture< T, cudaTextureType3D, cudaReadModeNormalizedFloat> t, float x, float, y, float z,float4 dx, float4 dy)	Texture Reference	NO	-
		texCubemapGrad(T *ptr, cudaTextureObject_t texObj, float x, float, y, float z,float4 dx, float4 dy); T texCubemapGrad(cudaTextureObject_t texObj, float x, float, y, float z,float4 dx, float4 dy)	Texture Object	NO	-
	texCubemapLod	__nv_tex_rmet_ret< T> texCubemapLod(texture< T, cudaTextureType3D, cudaReadModeElementType> t, float x, float, y, float z, float level); __nv_tex_rmnf_ret< T> texCubemapLod(texture< T, cudaTextureType3D, cudaReadModeNormalizedFloat> t, float x, float, y, float z, float level)	Texture Reference	NO	-

		texCubemapLod(T *ptr, cudaTextureObject_t texObj, float x, float, y, float z,float level); T texCubemapLod(cudaTextureObject_t texObj, float x, float, y, float z,float level)	Texture Object	NO	-
	texCubemapLayered	__nv_tex_rmet_ret texCubemapLayered(texture t, float x, float, y, float z, int level); __nv_tex_rmnf_ret texCubemapLayered(texture t, float x, float, y, float z, int level)	Texture Reference	NO	-
		texCubemapLayered(T *ptr,cudaTextureObject_t texObj,float x, float y, float z, int layer); T texCubemapLayered(cudaTextureObject_t texObj,float x, float y, float z, int layer)	Texture Object	NO	-
	texCubemapGrad	__nv_tex_rmet_ret< T> texCubemapGrad(texture< T, cudaTextureType3D, cudaReadModeElementType> t, float x, float, y, float z, int level); __nv_tex_rmnf_ret< T> texCubemapGrad(texture< T, cudaTextureType3D, cudaReadModeNormalizedFloat> t, float x, float, y, float z, int level)	Texture Reference	NO	-
		texCubemapGrad(T *ptr, cudaTextureObject_t texObj, float x, float, y, float z, int level); T texCubemapGrad(cudaTextureObject_t texObj, float x, float, y, float z, int level)	Texture Object	NO	-
	texCubemapLayeredLod	__nv_tex_rmet_ret< T> texCubemapLayeredLod(texture< T, cudaTextureType3D, cudaReadModeElementType> t, float x, float, y, float z, int layer, float level); __nv_tex_rmnf_ret< T> texCubemapLayeredLod(texture< T, cudaTextureType3D, cudaReadModeNormalizedFloat> t, float x, float, y, float z, int layer, float level)	Texture Reference	NO	-

		<code>texCubemapLayeredLod(T *ptr,cudaTextureObject_t texObj, float x, float y, float z,int layer, float level); T texCubemapLayeredLod(cudaTextureObject_t texObj, float x, float y, float z,int layer, float level)</code>	Texture Object	NO	-
--	--	--	-------------------	----	---