**视频通讯画面裁剪**

### 视频格式简介

1. 视频常用格式

BGRA：包含蓝色（B）、绿色（G）、红色（R）三个颜色通道，A（Alpha）代表透明度，BGR每个通道区间[0~255]，A通道区间[0~1.0]，可通过三路通道混合来获取不同颜色。类似格式有RGBA、AGBR等，元素一样，只是存储方式有区别。

YUV：包含明亮度（Y）、色度（UV），Y分量是完整的一帧图像，由于它本身没带色彩，所以是灰色的视频帧，而UV分量则包含色彩和饱和度，用来指定像素的颜色，常见格式：Y420、i420、NV12。区间？

按照格式大体划分2类：

1. Planar：连续像素点存储Y后，紧接着U，最后是V

2. Packed：连续像素点的Y、U、V交叉存储

1. 视频存储方式

BGRA：默认是32bit，每个通道是8bit，类似8 + 8 + 8 + 8，可以看做是一个int32位的数组存储。格式如下

|  |  |  |  |
| --- | --- | --- | --- |
| B1 | G1 | R1 | A1 |
| B2 | G2 | R2 | A2 |
| B3 | G3 | R3 | A3 |
| B4 | G4 | R4 | A4 |
| B5 | G5 | R5 | A5 |
| B6 | G6 | R6 | A6 |
| B7 | G7 | R7 | A7 |
| B8 | G8 | R8 | A8 |

i420：Y、U、V分量分别在不同平面上，每4个Y分量对应一个U和一个V。格式如下，简单例子：Y1，Y2，Y9，Y10，对应U1，V1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 | Y8 |
| Y9 | Y10 | Y11 | Y12 | Y13 | Y14 | Y15 | Y16 |
| Y17 | Y18 | Y19 | Y20 | Y21 | Y22 | Y23 | Y24 |
| Y25 | Y26 | Y27 | Y28 | Y29 | Y30 | Y31 | Y32 |
| U1 | U2 | U3 | U4 | U5 | U6 | U7 | U8 |
| V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 |

NV12：Y、UV分别在不同平面上，每4个Y分量对应一个UV。格式如下，简单例子：Y1，Y2，Y9，Y10，对应U1V1，而NV21只是U1V1排布改为V1U1而已

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 | Y8 |
| Y9 | Y10 | Y11 | Y12 | Y13 | Y14 | Y15 | Y16 |
| Y17 | Y18 | Y19 | Y20 | Y21 | Y22 | Y23 | Y24 |
| Y25 | Y26 | Y27 | Y28 | Y29 | Y30 | Y31 | Y32 |
| U1 | V1 | U2 | V2 | U3 | V3 | U4 | V4 |
| U5 | V5 | U6 | V6 | U7 | V7 | U8 | V8 |

1. 视频格式内存计算

BGRA：内存 = width \* height。

i420：由于4个Y对应一个U，一个V，Y = width \* height，U = Y / 4，V = Y / 4（通俗讲就是U和V的宽高是Y的宽高一半），然后i420内存 = Y + Y / 4 + Y / 4 = 3 / 2 \* Y。

NV12，NV21：存储大小跟i420基本一致，只是UV排布顺序有些区别而已

### iOS平台

1. iOS视频格式命名

kCVPixelFormatType\_{长度|序列}{颜色空间}{Planar|BiPlanar}{VideoRange|FullRange}

Planar: 平面；BiPlanar：双平面

平面／双平面主要应用在yuv上。uv分开存储的为Planar，反之是BiPlanar。所以，kCVPixelFormatType\_420YpCbCr8PlanarFullRange是420p，kCVPixelFormatType\_420YpCbCr8BiPlanarFullRange是nv12.

VideoRange和FullRange的区别在于数值的范围，FullRange比VideoRange大一些，颜色也更丰富一些，之前推流预览iOS有泛白现象，有2个原因引起：1.取的videorange导致颜色不够饱和 2.渲染的纹理类型精度不够。

如果没有指定颜色范围，默认都是FullRange。但有一个除外：kCVPixelFormatType\_420YpCbCr8Planar。因为有一个kCVPixelFormatType\_420YpCbCr8PlanarFullRange定义，所以kCVPixelFormatType\_420YpCbCr8Planar理论上应该是VideoRange (参考链接）

颜色空间对应的就是它在内存中的顺序。比如kCVPixelFormatType\_32BGRA，内存中的顺序是 B G R A B G R A...。

有一些特别的，比如kCVPixelFormatType\_16BE555，这里需要用BE或LE指定字节顺序。

有的颜色空间后面还带有一个数字，用于表示bit长度。

1. iOS平台裁剪
2. 通讯流程



1. 裁剪流程
   1. GPU转CPU方式裁剪：优点是适配性高，确定是CPU浮动较大并且占用高

#pragma mark Crop

/\* GPU crop -> CPU \*/

- (void \*) cropSampleBufferByHardware:(void \*) samplebuffer

{

/\* 计算当前图像如果类似16:9比例，不处理，否则计算裁剪后的区域 \*/

CMSampleBufferRef buffer = (CMSampleBufferRef) samplebuffer;

CVImageBufferRef imageBuffer = CMSampleBufferGetImageBuffer( buffer );

int frameWidth = (int) CVPixelBufferGetWidth( imageBuffer );

int frameHeight = (int) CVPixelBufferGetHeight( imageBuffer );

BOOL isPortrait = frameWidth > frameHeight ? false : true;

CGFloat standardScale = isPortrait ? (9.0 / 16.0) : (16.0 / 9.0);

CGFloat currentScale = frameWidth \* 1.0 / frameHeight;

CGFloat scaleError = 0.1;

CGFloat videoX = 0;

CGFloat videoY = 0;

CGFloat videoWidth = frameWidth;

CGFloat videoHeight = frameHeight;

if ( currentScale - standardScale > scaleError )

{

if ( isPortrait )

{

videoWidth = frameHeight \* 9 / 16;

}else {

videoWidth = frameHeight \* 16 / 9;

}

videoX = (frameWidth - videoWidth) / 2;

}else if ( standardScale - currentScale > scaleError )

{

if ( isPortrait )

{

videoHeight = frameWidth \* 16 / 9;

}else {

videoHeight = frameWidth \* 9 / 16;

}

videoY = (frameHeight - videoHeight) / 2;

}else {

return(nil);

}

/\* 创建GPU的上下文 \*/

CGRect cropRect = CGRectMake( videoX, videoY, videoWidth, videoHeight );

static CIContext \*ciContext = nil;

if ( ciContext == nil )

{

ciContext = [CIContext contextWithEAGLContext:context options:nil];

}

OSStatus status;

static CVPixelBufferRef pixbuffer = NULL;

static CMVideoFormatDescriptionRef videoInfo = NULL;

/\* 创建存储新的裁剪后的pixbuffer（用来生成编码专用samplebuffer），格式跟来源格式基本保持一致，理论上讲也可以是平台支持的任意格式 \*/

if ( pixbuffer == NULL )

{

NSDictionary \*options = [NSDictionary dictionaryWithObjectsAndKeys:

[NSNumber numberWithInt:videoWidth], kCVPixelBufferWidthKey,

[NSNumber numberWithInt:videoHeight], kCVPixelBufferHeightKey,

@ {}, kCVPixelBufferIOSurfacePropertiesKey,

nil];

status = CVPixelBufferCreate( kCFAllocatorSystemDefault, videoWidth, videoHeight, kCVPixelFormatType\_420YpCbCr8BiPlanarFullRange, (\_\_bridge CFDictionaryRef) options, &pixbuffer );

if ( status != noErr )

{

NSLog( @"Crop CVPixelBufferCreate error %d", (int) status );

return(NULL);

}

}

/\* 获取源数据的CIIimage数据 \*/

CIImage \*ciImage = [CIImage imageWithCVImageBuffer:imageBuffer];

/\* 裁剪为16:9比例的CIImage \*/

ciImage = [ciImage imageByCroppingToRect:cropRect];

ciImage = [ciImage imageByApplyingTransform:CGAffineTransformMakeTranslation( -videoX, -videoY )];

/\* 采用上下文将裁剪后的数据转换到新的pixbuffer缓存里面（GPU-）CPU） \*/

[ciContext render:ciImage toCVPixelBuffer:pixbuffer];

CMSampleTimingInfo sampleTime = {

.duration = CMSampleBufferGetDuration( buffer ),

.presentationTimeStamp = CMSampleBufferGetPresentationTimeStamp( buffer ),

.decodeTimeStamp = CMSampleBufferGetDecodeTimeStamp( buffer )

};

if ( videoInfo == NULL )

{

status = CMVideoFormatDescriptionCreateForImageBuffer( kCFAllocatorDefault, pixbuffer, &videoInfo );

if ( status != 0 )

NSLog( @"Crop CMVideoFormatDescriptionCreateForImageBuffer error %d", (int) status );

}

/\* 根据pixbuffer转换为samplebuffer（编码专用格式） \*/

CMSampleBufferRef cropBuffer = nil;

status = CMSampleBufferCreateForImageBuffer( kCFAllocatorDefault, pixbuffer, true, NULL, NULL, videoInfo, &sampleTime, &cropBuffer );

if ( status != 0 )

NSLog( @"Crop CMSampleBufferCreateForImageBuffer error %d", (int) status );

return(cropBuffer);

}

* 1. CPU裁剪

- (void \*) cropSampleBufferBySoftware:(void \*) buffer

{

if ( buffer == nil )

return(nil);

/\* 计算当前图像如果类似16:9比例，不处理，否则计算裁剪后的区域 \*/

CMSampleBufferRef sampleBuffer = (CMSampleBufferRef) buffer;

CVImageBufferRef imageBuffer = CMSampleBufferGetImageBuffer( (CMSampleBufferRef) sampleBuffer );

/\* YUV 420 Rule \*/

int frameYWidth = (int) CVPixelBufferGetWidthOfPlane( imageBuffer, 0 );

int frameHeight = (int) CVPixelBufferGetHeight( imageBuffer );

BOOL isPortrait = frameYWidth > frameHeight ? false : true;

CGFloat standardScale = isPortrait ? (9.0 / 16.0) : (16.0 / 9.0);

CGFloat currentScale = frameYWidth \* 1.0 / frameHeight;

CGFloat scaleError = 0.1;

CGFloat videoX = 0;

CGFloat videoY = 0;

CGFloat videoWidth = frameYWidth;

CGFloat videoHeight = frameHeight;

if ( currentScale - standardScale > scaleError )

{

if ( isPortrait )

{

videoWidth = frameHeight \* 9 / 16;

}else {

videoWidth = frameHeight \* 16 / 9;

}

videoX = (frameYWidth - videoWidth) / 2;

}else if ( standardScale - currentScale > scaleError )

{

if ( isPortrait )

{

videoHeight = frameYWidth \* 16 / 9;

}else {

videoHeight = frameYWidth \* 9 / 16;

}

videoY = (frameHeight - videoHeight) / 2;

}else {

return(nil);

}

OSType pixelformat = CVPixelBufferGetPixelFormatType( imageBuffer );

CMSampleBufferRef cropBuffer = nil;

/\* 根据pixbuffer的视频格式来区分CPU内存拷贝方式 \*/

switch ( pixelformat )

{

case kCVPixelFormatType\_420YpCbCr8BiPlanarFullRange:

case kCVPixelFormatType\_420YpCbCr8BiPlanarVideoRange:

{

cropBuffer = [self getCropedPixbufferWithNV12:sampleBuffer cropRect:CGRectMake( videoX, videoY, videoWidth, videoHeight )];

}

break;

case kCVPixelFormatType\_32BGRA:

{

cropBuffer = [self getCropedPixbufferWithBGRA:sampleBuffer cropRect:CGRectMake( videoX, videoY, videoWidth, videoHeight )];

}

break;

case kCVPixelFormatType\_420YpCbCr8Planar:

{

cropBuffer = [self getCropedPixbufferWithYUV420:sampleBuffer cropRect:CGRectMake( videoX, videoY, videoWidth, videoHeight )];

}

break;

default:

break;

}

return(cropBuffer);

}

// NV12格式的裁剪

- (CMSampleBufferRef) getCropedPixbufferWithNV12:(CMSampleBufferRef) samplebuffer cropRect:(CGRect) croprect

{

OSStatus status;

CVImageBufferRef sourcebuffer = CMSampleBufferGetImageBuffer( samplebuffer );

/\* Get information about the image \*/

size\_t bytesPerRow = CVPixelBufferGetBytesPerRowOfPlane( sourcebuffer, 0 );

size\_t bytesPerRowUV = CVPixelBufferGetBytesPerRowOfPlane( sourcebuffer, 1 );

size\_t width = CVPixelBufferGetWidth( sourcebuffer );

NSInteger bytesPerPixel = bytesPerRow / width;

int tmpX = (int) croprect.origin.x;

int tmpY = (int) croprect.origin.y;

if ( tmpX % 2 != 0 )

tmpX += 1;

NSInteger baseAddressStart = (int) tmpY \* bytesPerRow + bytesPerPixel \* (int) tmpX;

static NSInteger lastAddressStart = 0;

lastAddressStart = baseAddressStart;

NSLog( @"getCropedPixbufferWithNV12 videoWidth %f videoheight %f baseAddressStart %ld", croprect.size.width, croprect.size.height, baseAddressStart );

/\* pixbuffer 与 videoInfo 只有位置变换或者切换分辨率或者相机重启时需要更新，其余情况不需要 \*/

static CVPixelBufferRef pixbuffer = NULL;

static CMVideoFormatDescriptionRef videoInfo = NULL;

/\* x,y changed need to reset pixbuffer and videoinfo \*/

if ( lastAddressStart != baseAddressStart )

{

if ( pixbuffer != NULL )

{

CVPixelBufferRelease( pixbuffer );

pixbuffer = NULL;

}

if ( videoInfo != NULL )

{

CFRelease( videoInfo );

videoInfo = NULL;

}

}

if ( pixbuffer == NULL )

{

NSDictionary \*options = [NSDictionary dictionaryWithObjectsAndKeys:

[NSNumber numberWithBool : YES], kCVPixelBufferCGImageCompatibilityKey,

[NSNumber numberWithBool : YES], kCVPixelBufferCGBitmapContextCompatibilityKey,

[NSNumber numberWithInt : croprect.size.width], kCVPixelBufferWidthKey,

[NSNumber numberWithInt : croprect.size.height], kCVPixelBufferHeightKey,

@ {}, kCVPixelBufferIOSurfacePropertiesKey,

nil];

status = CVPixelBufferCreate( kCFAllocatorDefault, croprect.size.width, croprect.size.height, kCVPixelFormatType\_420YpCbCr8BiPlanarFullRange, (\_\_bridge CFDictionaryRef) options, &pixbuffer );

if ( status != 0 )

{

NSLog( @"Crop CVPixelBufferCreateWithBytes error %d", (int) status );

return(NULL);

}

}

// 按照NV12格式将内存进行逐行copy到新的pixbuffer

/\* Lock the image buffer \*/

CVPixelBufferLockBaseAddress( sourcebuffer, 0 );

CVPixelBufferLockBaseAddress( pixbuffer, 0 );

uint8\_t \*yPlane = (uint8\_t \*) CVPixelBufferGetBaseAddressOfPlane( sourcebuffer, 0 );

uint8\_t \*yDestPlane = (uint8\_t \*) CVPixelBufferGetBaseAddressOfPlane( pixbuffer, 0 );

int yBytesPerRowOfPixbuffer = (int) CVPixelBufferGetBytesPerRowOfPlane( pixbuffer, 0 );

int videoHeightY = (int) CVPixelBufferGetHeightOfPlane( pixbuffer, 0 );

long offset = baseAddressStart;

for ( int index = 0; index < videoHeightY; index++ )

{

memcpy( yDestPlane + (long) yBytesPerRowOfPixbuffer \* index, yPlane + offset, yBytesPerRowOfPixbuffer );

offset += bytesPerRow;

}

uint8\_t \*uvPlane = (uint8\_t \*) CVPixelBufferGetBaseAddressOfPlane( sourcebuffer, 1 );

uint8\_t \*uvDestPlane = (uint8\_t \*) CVPixelBufferGetBaseAddressOfPlane( pixbuffer, 1 );

int uvBytesPerRowOfPixbuffer = (int) CVPixelBufferGetBytesPerRowOfPlane( pixbuffer, 1 );

int videoHeightUV = (int) CVPixelBufferGetHeightOfPlane( pixbuffer, 1 );

offset = baseAddressStart;

for ( int index = 0; index < videoHeightUV; index++ )

{

memcpy( uvDestPlane + (long) uvBytesPerRowOfPixbuffer \* index, uvPlane + offset, uvBytesPerRowOfPixbuffer );

offset += bytesPerRowUV;

}

CVPixelBufferUnlockBaseAddress( pixbuffer, 0 );

CVPixelBufferUnlockBaseAddress( sourcebuffer, 0 );

CMSampleTimingInfo sampleTime = {

.duration = CMSampleBufferGetDuration( samplebuffer ),

.presentationTimeStamp = CMSampleBufferGetPresentationTimeStamp( samplebuffer ),

.decodeTimeStamp = CMSampleBufferGetDecodeTimeStamp( samplebuffer )

};

if ( videoInfo == NULL )

{

status = CMVideoFormatDescriptionCreateForImageBuffer( kCFAllocatorDefault, pixbuffer, &videoInfo );

if ( status != 0 )

NSLog( @"Crop CMVideoFormatDescriptionCreateForImageBuffer error %d", (int) status );

}

CMSampleBufferRef cropBuffer = NULL;

status = CMSampleBufferCreateForImageBuffer( kCFAllocatorDefault, pixbuffer, true, NULL, NULL, videoInfo, &sampleTime, &cropBuffer );

if ( status != 0 )

NSLog( @"Crop CMSampleBufferCreateForImageBuffer error %d", (int) status );

lastAddressStart = baseAddressStart;

return(cropBuffer);

}