前 言

此篇文章分成五個部份

(1) YUV(YCBCR)轉成 RGB - YUV420,YUV422 及 YUV444 轉成 RGB

(2) RGB raster data 顯示 - 使用 Win API StretchDIBits

(3) RGB raster data 存檔 - 存成 BMP 檔

(4) YUV 轉成 RGB 並且存取 BMP 檔實例

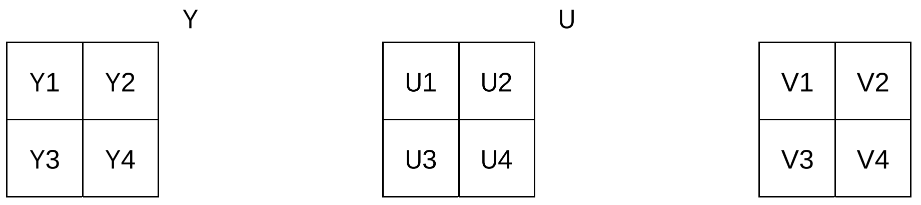
(5) 參考文獻

將 YUV raster data 顯示在螢幕上，先要轉成 RGB，再配合上 Header 呼叫 Win32 API 函式，即能將 YUV raster data 顯示出來。

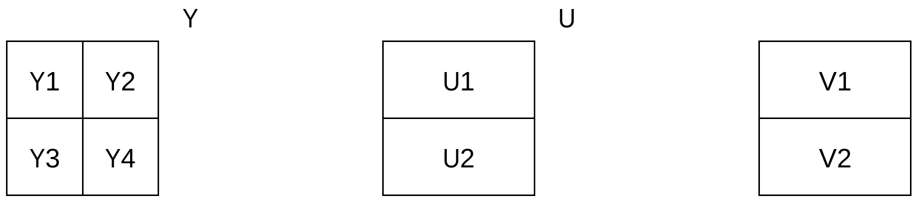
1.YUV(YCBCR)轉成 RGB

說明轉換之前，先瞭解 YUV 的三種格式，分別為 YUV420、YUV422、YUV44

(1) YUV444：沒有 vertical subsampling 及 horizontal subsampling，Y:U:V = 4:4:4

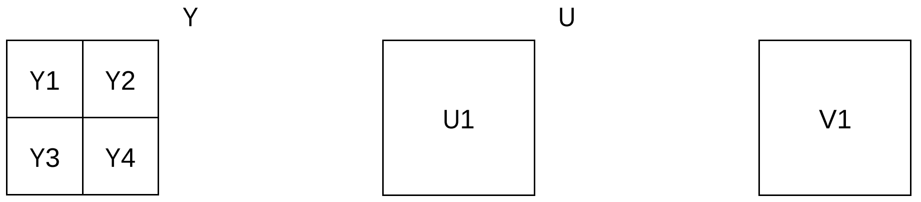


(2) YUV422：有 horizontal subsampling，沒有 vertical subsampling，Y:U:V = 4:2:2



Y1 及 Y2 共用 U1 及 V1；Y3 及 Y4 共用 U2 及 V2

(3) YUV420：有 vertical subsampling 亦有 horizontal subsampling，**Y:U:V = 4:1:1**



Y1~Y4 共用 U1 及 V1

瞭解 YUV subsampling 的格式之後，接下來說明 YUV 與 RGB 的轉換關係，在一般 Video 中，YUV 轉成 RGB 的定義：

$$\begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} = \begin{bmatrix} 1.16438 & 0 & 1.59603 \\ 1.16438 & -0.39176 & -0.81297 \\ 1.16438 & 2.01723 & 0 \end{bmatrix} \begin{bmatrix} Y-16 \\ U-128 \\ V-128 \end{bmatrix}$$

Y、U 及 V 的值在 0~255 之間，經過轉換後，可能小於 0 或大於 255，爲了確保 R、G 及 B 的值亦在 0~255 之間，可以在轉換處理

```
IF R/G/B is greater than 255 THEN
    R/G/B := 255
IF R/G/B is less than 0 THEN
    R/G/B := 0
```

以上的方式十分簡單明瞭，但速度慢。從一位 Hao Pan 研究 computer vision, multimedia 及 signal processing 的網站中取得 Fast conversion 的方法，利用「Clip table」來取代以上的判斷式，根據實驗結果，建立一個表格：

(1) 值落在-384 ~ -1 之間，值爲 0

(2) 值落在 0 ~ 255 之間，值爲原本的值

(3) 值落在 256 ~ 639 之間，值爲 255

目前先提出這樣方法，在之後的內容中會實作出來。

從 YUV 轉成 RGB 的關係式中，用到浮點數而使得程式速度降低，爲了加快速度，根據 Hao Pan 的方法，將浮點數同乘 65536，即 2^16(右移 16bit)，關係式變爲：

$$\begin{bmatrix} \mathrm{R} \\ \mathrm{G} \\ \mathrm{B} \end{bmatrix} = \frac{1}{65536}\begin{bmatrix} 76309 & 0 & 104597 \\ 76309 & -25675 & -53279 \\ 76309 & 132201 & 0 \end{bmatrix}\begin{bmatrix} Y-16 \\ U-128 \\ V-128 \end{bmatrix}$$

這個關係式搭配著「Clip table」使用。

不同 YUV Subsampling 的格式轉成 RGB 及 Clip table 的完整程式，提供在此，轉換的方法建議從程式中去學習。

說明：建立 Clip table 及 YUV to RGB 的 Conversion table

```
static long int crv_tab[256];
static long int cbu_tab[256];
static long int cgu_tab[256];
static long int cgv_tab[256];
static long int tab_76309[256];
static unsigned char clp[1024];

void InitConvtTbl()
{
    long int crv,cbu,cgu,cgv;
    int i,ind;

    crv = 104597; cbu = 132201;
    cgu = 25675;   cgv = 53279;

    for (i = 0; i < 256; i++) {
        crv_tab[i] = (i-128) * crv;
        cbu_tab[i] = (i-128) * cbu;
        cgu_tab[i] = (i-128) * cgu;
        cgv_tab[i] = (i-128) * cgv;
        tab_76309[i] = 76309*(i-16);
    }

    for (i=0; i<384; i++)
        clp[i] =0;
```

```
		ind=384;
		for (i=0;i<256; i++)
			clp[ind++]=i;
		ind=640;
		for (i=0;i<384;i++)
			clp[ind++]=255;
	}
```

說明：

轉換 YUV420 成 RGB

```
// Input : YUV raster data,YY...YYY,UU...UUU,VV...VVV
// Output: RGB buffer,R,G,B,R,G,B,...,R,G,B
// Parameter :
// src : pointer to YUV raster data
// dst_ori : pointer to RGB buffer
// width : image width
// height : image height
void YUV420toRGB(unsigned char *src, unsigned char *dst_ori,
                                        int width,int height)
{
	unsigned char *src0;
	unsigned char *src1;
	unsigned char *src2;
	int y1,y2,u,v;
	unsigned char *py1,*py2;
	int i,j, c1, c2, c3, c4;
	unsigned char *d1, *d2, *d3;

	//Initialization
	src0=src;
	src1=src+width*height;
	src2=src+width*height+width*height/4;

	py1=src0;
	py2=py1+width;
	d1=dst_ori;
	d2=d1+3*width;
	for (j = 0; j < height; j += 2) {
		for (i = 0; i < width; i += 2) {

			u = *src1++;
			v = *src2++;

			c1 = crv_tab[v];
			c2 = cgu_tab[u];
			c3 = cgv_tab[v];
			c4 = cbu_tab[u];

			//up-left
			y1 = tab_76309[*py1++];
			*d1++ = clp[384+((y1 + c1)>>16)];
			*d1++ = clp[384+((y1 - c2 - c3)>>16)];
			*d1++ = clp[384+((y1 + c4)>>16)];
```

```
                //down-left
                y2 = tab_76309[*py2++];
                *d2++ = clp[384+((y2 + c1)>>16)];
                *d2++ = clp[384+((y2 - c2 - c3)>>16)];
                *d2++ = clp[384+((y2 + c4)>>16)];

                //up-right
                y1 = tab_76309[*py1++];
                *d1++ = clp[384+((y1 + c1)>>16)];
                *d1++ = clp[384+((y1 - c2 - c3)>>16)];
                *d1++ = clp[384+((y1 + c4)>>16)];

                //down-right
                y2 = tab_76309[*py2++];
                *d2++ = clp[384+((y2 + c1)>>16)];
                *d2++ = clp[384+((y2 - c2 - c3)>>16)];
                *d2++ = clp[384+((y2 + c4)>>16)];
            }
            d1 += 3*width;
            d2 += 3*width;
            py1+=    width;
            py2+=    width;
        }

}
```

轉換 YUV422 成 RGB

```
// Input : YUV raster,YY...YYY,UU...UUU,VV...VVV
// Output: RGB buffer, R,G,B,R,G,B,...,R,G,B
// Parameter :
// src : pointer to YUV raster data
// dst_ori : pointer to RGB buffer
// width : image width
// height : image height
void YUV422toRGB(unsigned char *src, unsigned char *dst_ori, int width, int height)
{
        unsigned char *src0;
        unsigned char *src1;
        unsigned char *src2;
        unsigned char *dst;
        unsigned char *py1,*py2;
        unsigned char *d1, *d2, *d3;
        int y1,u,v;
        int i,j, c1, c2, c3, c4;


        //Initialization
        src0=src;
        src1=src+width*height;
        src2=src+width*height+width*height/2;

        dst=dst_ori;
```

```
	for (j = 0; j < height; j += 1) {
		for (i = 0; i < width; i += 2) {

			u = *src1++;
			v = *src2++;

			c1 = crv_tab[v];
			c2 = cgu_tab[u];
			c3 = cgv_tab[v];
			c4 = cbu_tab[u];

			y1 = tab_76309[*src0++];
			*dst++ = clp[384+((y1 + c1)>>16)];
			*dst++ = clp[384+((y1 - c2 - c3)>>16)];
			*dst++ = clp[384+((y1 + c4)>>16)];

			y1 = tab_76309[*src0++];
			*dst++ = clp[384+((y1 + c1)>>16)];
			*dst++ = clp[384+((y1 - c2 - c3)>>16)];
			*dst++ = clp[384+((y1 + c4)>>16)];
		}
	}

}
```

轉換 YUV444 成 RGB

```
// Input : YUV raster,YY...YYY,UU...UUU,VV...VVV
// Output: RGB buffer, R,G,B,R,G,B,...,R,G,B
// Parameter :
// src : pointer to YUV raster data
// dst_ori : pointer to RGB buffer
// width : image width
// height : image height
void YUV444toRGB(unsigned char *src, unsigned char *dst_ori, int width, int height)
{
	unsigned char *src0;
	unsigned char *src1;
	unsigned char *src2;
	unsigned char *dst;
	int y1,u,v;
	int i,j, c1, c2, c3, c4;


	//Initialization
	src0=src;
	src1=src+width*height;
	src2=src+width*height*2;

	dst=dst_ori;

	for (j = 0; j < height; j += 1) {
		for (i = 0; i < width; i ++ ) {

			u = *src1++;
```

```
                        v = *src2++;

                        c1 = crv_tab[v];
                        c2 = cgu_tab[u];
                        c3 = cgv_tab[v];
                        c4 = cbu_tab[u];

                        y1 = tab_76309[*src0++];
                        *dst++ = clp[384+((y1 + c1)>>16)];
                        *dst++ = clp[384+((y1 - c2 - c3)>>16)];
                        *dst++ = clp[384+((y1 + c4)>>16)];

                }
        }

}
```

2.RGB raster data 顯示

目前有 RGB raster data，要如何讓電腦來顯示呢？很簡單，讓電腦依據所提供的資訊(Information)來建構出的影像，資訊包括寬(width)、高(height)、壓縮型態(Compression type)、影像大小(Size of Image)、...等，一般稱這個資訊爲「標頭(Header)」。利用 Window Programming 來處理顯示的動作，在 Window Bitmaps 有兩個地方要注意：

(1) raster data 的放置順序是 Blue、Green、Red(BGR)

(2) raster data 儲存資料的方式是由影像的底部至頂部，所以 raster data 的第一列，爲影像的最後一列
[left to right,bottom to top]

以實際的程式來說明：

```
BITMAPINFOHEADER bm;

bm.biSize = sizeof(BITMAPINFOHEADER);
bm.biWidth = FrameWidth;
bm.biHeight = FrameHeight;
bm.biPlanes = 1;
bm.biBitCount = 24;
bm.biCompression = BI_RGB;
bm.biSizeImage = 3 * FrameWidth * FrameHeight;
bm.biXPelsPerMeter = 0;
bm.biYPelsPerMeter = 0;
bm.biClrUsed = 0;
bm.biClrImportant = 0;
```

BITMAPINFOHEADER 爲 DIB header，在呼叫 StretchDIBits API 需要用到，可以參考在 MSDN 中一篇名爲 【 DIBs and Their Use 】的文章，內有詳細的說明，在此不多做解釋。以上是呼叫 StretchDIBits 的方式：

```
StretchDIBits(hdc,0,0,FrameWidth,FrameHeight,
                     0,0,FrameWidth,FrameHeight,
                    (LPVOID)RGB,
                    &bm,
                    DIB_RGB_COLORS,
                    SRCCOPY);
```

Note：RGB 是儲存 RGB raster data 的 buffer，data 的順序爲 B、G、R，且爲從底部到頂部。
呼叫的參數在【 DIBs and Their Use 】有詳細說明，或是參考 MSDN 中對 StretchDIBits 的描述。
若有 YUV raster data 時，要轉成正確 bitmaps raster data 順序，此時更改以上提供的三個 YUV to RGB 函式，更改如下

```
// Input : YUV raster,YY...YYY,UU...UUU,VV...VVV
// Output: RGB buffer, B,G,R,B,G,R...,B,R,G, [left to right,bottom to top]
// Parameter :
// src : pointer to YUV raster data
// dst_ori : pointer to RGB buffer
// width : image width
// height : image height
void YUV420toRGB(unsigned char *src, unsigned char *dst_ori, int width,int height)
{
     unsigned char *src0;
     unsigned char *src1;
     unsigned char *src2;
     int y1,y2,u,v;
     unsigned char *py1,*py2;
     int i,j, c1, c2, c3, c4;
```

```
        unsigned char *d1, *d2;

        //Initialization
        src0=src;
        src1=src+width*height;
        src2=src+width*height+width*height/4;

        py1=src0;
        py2=py1+width;
        d1=dst_ori + width*height*3 - width*3 ;
        d2=d1 - 3*width;

        for (j = 0; j < height; j += 2) {
                for (i = 0; i < width; i += 2) {

                        u = *src1++;
                        v = *src2++;

                        c1 = crv_tab[v];
                        c2 = cgu_tab[u];
                        c3 = cgv_tab[v];
                        c4 = cbu_tab[u];

                        //up-left
                        y1 = tab_76309[*py1++];
                        *d1++ = clp[384+((y1 + c4)>>16)];          // Blue
                        *d1++ = clp[384+((y1 - c2 - c3)>>16)];     // Green
                        *d1++ = clp[384+((y1 + c1)>>16)];          // Red

                        //down-left
                        y2 = tab_76309[*py2++];
                     *d2++ = clp[384+((y2 + c4)>>16)];             // Blue
                        *d2++ = clp[384+((y2 - c2 - c3)>>16)];     // Green
                        *d2++ = clp[384+((y2 + c1)>>16)];          // Red

                        //up-right
                        y1 = tab_76309[*py1++];
                        *d1++ = clp[384+((y1 + c4)>>16)];          // Blue
                        *d1++ = clp[384+((y1 - c2 - c3)>>16)];     // Green
                        *d1++ = clp[384+((y1 + c1)>>16)];          // Red

                        //down-right
                        y2 = tab_76309[*py2++];
                        *d2++ = clp[384+((y2 + c4)>>16)];          // Blue
                        *d2++ = clp[384+((y2 - c2 - c3)>>16)];     // Green
                        *d2++ = clp[384+((y2 + c1)>>16)];          // Red

                }
                d1 -= 9*width;
                d2 -= 9*width;
                py1+=     width;
                py2+=     width;
        }

}
```

// Input : YUV raster,YY...YYY,UU...UUU,VV...VVV

// Output: RGB buffer, B,G,R,B,G,R...,B,R,G, [left to right,bottom to top]

```
// Parameter :
// src : pointer to YUV raster data
// dst_ori : pointer to RGB buffer
// width : image width
// height : image height
void YUV422toRGB(unsigned char *src, unsigned char *dst_ori, int width, int height)
{
        unsigned char *src0;
        unsigned char *src1;
        unsigned char *src2;
        unsigned char *dst;
        int y1,u,v;
        int i,j, c1, c2, c3, c4;


        //Initialization
        src0=src;
        src1=src+width*height;
        src2=src+width*height+width*height/2;

        dst= dst_ori+ width*height*3 - width*3;

        for (j = 0; j < height; j += 1) {
                for (i = 0; i < width; i += 2) {

                        u = *src1++;
                        v = *src2++;

                        c1 = crv_tab[v];
                        c2 = cgu_tab[u];
                        c3 = cgv_tab[v];
                        c4 = cbu_tab[u];

                        y1 = tab_76309[*src0++];
                        *dst++ = clp[384+((y1 + c4)>>16)];              // Blue
                        *dst++ = clp[384+((y1 - c2 - c3)>>16)];         // Green
                        *dst++ = clp[384+((y1 + c1)>>16)];              // Red

                        y1 = tab_76309[*src0++];
                        *dst++ = clp[384+((y1 + c4)>>16)];              // Blue
                        *dst++ = clp[384+((y1 - c2 - c3)>>16)];         // Green
                        *dst++ = clp[384+((y1 + c1)>>16)];              // Red

                }
                dst -= width*6;
        }

}
```

```
// Input : YUV raster,YY...YYY,UU...UUU,VV...VVV
// Output: RGB buffer, B,G,R,B,G,R...,B,R,G, [left to right,bottom to top]
// Parameter :
// src : pointer to YUV raster data
// dst_ori : pointer to RGB buffer
// width : image width
// height : image height
void YUV444toRGB(unsigned char *src,
                                unsigned char *dst_ori,
                                int width, int height)
{
        unsigned char *src0;
        unsigned char *src1;
        unsigned char *src2;
        unsigned char *dst;
        int y1,u,v;
        int i,j, c1, c2, c3, c4;


        //Initialization
        src0=src;
        src1=src+width*height;
        src2=src+width*height*2;

        dst=dst_ori+ width*height*3 - width*3;

        for (j = 0; j < height; j += 1) {
                for (i = 0; i < width; i ++ ) {

                        u = *src1++;
                        v = *src2++;

                        c1 = crv_tab[v];
                        c2 = cgu_tab[u];
                        c3 = cgv_tab[v];
                        c4 = cbu_tab[u];

                         y1 = tab_76309[*src0++];
                        *dst++ = clp[384+((y1 + c4)>>16)];              // Blue
                        *dst++ = clp[384+((y1 - c2 - c3)>>16)];         // Green
                        *dst++ = clp[384+((y1 + c1)>>16)];              // Red
                }
                dst -= width * 6;
        }

}
```

3.RGB raster data 存檔

要將 RGB raster data 存成.BMP 的檔案，首要條件就是要清楚 BMP 檔案的格式，可以參考這一篇文章：
【DaubNET File Formats Collection BMP: http://www.daubnet.com/formats/BMP.html#RasterData】
BMP 檔案格式，主要分爲：

(1)Header
(2)InfoHeader
(3)Color Table if BitCount<=8
(4)Raster Data

一般來說，BitCount 大於 8，以 256 彩色來說，BitCount 爲 24。以程式來實作儲存 raster data 爲.BMP 檔：

```
BITMAPFILEHEADER bmheader;
BITMAPINFOHEADER bm;

bm.biSize = sizeof(BITMAPINFOHEADER);
bm.biWidth = FrameWidth;
bm.biHeight = FrameHeight;
bm.biPlanes = 1;
bm.biBitCount = 24;
bm.biCompression = BI_RGB;
bm.biSizeImage = 3 * FrameWidth * FrameHeight;
bm.biXPelsPerMeter = 0;
bm.biYPelsPerMeter = 0;
bm.biClrUsed = 0;
bm.biClrImportant = 0;

bmheader.bfType = 0x4D42;  // 0x42 = "B" 0x4d = "M"
bmheader.bfSize =   (DWORD) (sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) +
                              bm.biSizeImage );
bmheader.bfReserved1 = 0;
bmheader.bfReserved2 = 0;
bmheader.bfOffBits = (DWORD) (sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) );

fwrite(&bmheader,1,sizeof(BITMAPFILEHEADER),target_file);
fwrite(&bm,1,sizeof(BITMAPINFOHEADER),target_file);
fwrite(RGB_raster_data,1,FrameHeight*FrameWidth*3,target_file);
```

之處只是大約說明方法，要詳細的解說還是要查 MSDN 中去閱讀。

4.YUV 轉成 RGB 並且存取 BMP 檔實例

輸入檔案 YUV420.yuv 是從 QCIF sequences[http://trace.eas.asu.edu/yuv/qcif.html]網頁中的 carphone sequences，抓出第一個 Frame 的 sequence。QCIF 的寬爲 176，高爲 144。

```
#include <stdio.h>
#include <windows.h>
#define WIDTH       176
#define HEIGHT      144

long int crv_tab[256];
long int cbu_tab[256];
long int cgu_tab[256];
long int cgv_tab[256];
long int tab_76309[256];
unsigned char clp[1024];

void InitConvtTbl();
void YUV420toRGB(unsigned char *src,unsigned char *dst_ori,int width,int height);

int main()
{
	FILE *source,*target;
	unsigned char *YUV,*RGB;
	BITMAPFILEHEADER bmheader;
	BITMAPINFOHEADER bm;

	source = fopen("YUV420.yuv","rb");
	if ( source == NULL )
	{
		printf("Source file opens failure!\n");
		return 0;
	}

	target = fopen("carphone.bmp","wb");
	if ( target == NULL )
	{
		printf("Target file opens failure!\n");
		return 0;
	}

	YUV = (unsigned char *)malloc( sizeof(unsigned char) * WIDTH * HEIGHT * 3 / 2 );
	RGB = (unsigned char *)malloc( sizeof(unsigned char) * WIDTH * HEIGHT * 3      );

	memset( YUV , 0 , sizeof(unsigned char) * WIDTH * HEIGHT * 3 / 2 );
	memset( RGB , 0 , sizeof(unsigned char) * WIDTH * HEIGHT * 3      );

	fread( YUV , 1 , WIDTH * HEIGHT * 3 / 2 , source );

	InitConvtTbl();
	YUV420toRGB( YUV , RGB , WIDTH , HEIGHT );

	bm.biSize = sizeof(BITMAPINFOHEADER);
	bm.biWidth = WIDTH;
	bm.biHeight = HEIGHT;
	bm.biPlanes = 1;
	bm.biBitCount = 24;
	bm.biCompression = BI_RGB;
	bm.biSizeImage = 3 * WIDTH * HEIGHT;
	bm.biXPelsPerMeter = 0;
```

```
      bm.biYPelsPerMeter = 0;
      bm.biClrUsed = 0;
      bm.biClrImportant = 0;

      bmheader.bfType = 0x4D42;   // 0x42 = "B" 0x4d = "M"
     bmheader.bfSize =   (DWORD) (sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) +
                         bm.biSizeImage );
      bmheader.bfReserved1 = 0;
      bmheader.bfReserved2 = 0;
      bmheader.bfOffBits = (DWORD) (sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) );

      fwrite(&bmheader,sizeof(unsigned char),sizeof(BITMAPFILEHEADER),target);
      fwrite(&bm,sizeof(unsigned char),sizeof(BITMAPINFOHEADER),target);
      fwrite(RGB,sizeof(unsigned char),WIDTH*HEIGHT*3,target);

      free(RGB);
      free(YUV);
      RGB = NULL;
      YUV = NULL;
      fclose(source);
      fclose(target);
      return 0;
}


void InitConvtTbl()
{
    long int crv,cbu,cgu,cgv;
    int i,ind;

    crv = 104597; cbu = 132201;   /* fra matrise i global.h */
    cgu = 25675;   cgv = 53279;

    for (i = 0; i < 256; i++) {
        crv_tab[i] = (i-128) * crv;
        cbu_tab[i] = (i-128) * cbu;
        cgu_tab[i] = (i-128) * cgu;
        cgv_tab[i] = (i-128) * cgv;
        tab_76309[i] = 76309*(i-16);
    }

    for (i=0; i<384; i++)
         clp[i] =0;
    ind=384;
    for (i=0;i<256; i++)
          clp[ind++]=i;
    ind=640;
    for (i=0;i<384;i++)
          clp[ind++]=255;
}

void YUV420toRGB(unsigned char *src,unsigned char *dst_ori,int width,int height)
{
      unsigned char *src0;
      unsigned char *src1;
      unsigned char *src2;
      int y1,y2,u,v;
      unsigned char *py1,*py2;
      int i,j, c1, c2, c3, c4;
      unsigned char *d1, *d2;

      //Initialization
      src0=src;
      src1=src+width*height;
      src2=src+width*height+width*height/4;

      py1=src0;
      py2=py1+width;
      d1=dst_ori + width*height*3 - width*3 ;
      d2=d1 - 3*width;

      for (j = 0; j < height; j += 2) {
          for (i = 0; i < width; i += 2) {

              u = *src1++;
              v = *src2++;

              c1 = crv_tab[v];
              c2 = cgu_tab[u];
              c3 = cgv_tab[v];
              c4 = cbu_tab[u];

              //up-left
               y1 = tab_76309[*py1++];
              *d1++ = clp[384+((y1 + c4)>>16)];         // Blue
              *d1++ = clp[384+((y1 - c2 - c3)>>16)];    // Green
              *d1++ = clp[384+((y1 + c1)>>16)];         // Red

              //down-left
              y2 = tab_76309[*py2++];
              *d2++ = clp[384+((y2 + c4)>>16)];         // Blue
              *d2++ = clp[384+((y2 - c2 - c3)>>16)];    // Green
              *d2++ = clp[384+((y2 + c1)>>16)];         // Red

              //up-right
              y1 = tab_76309[*py1++];
              *d1++ = clp[384+((y1 + c4)>>16)];         // Blue
              *d1++ = clp[384+((y1 - c2 - c3)>>16)];    // Green
              *d1++ = clp[384+((y1 + c1)>>16)];         // Red

              //down-right
              y2 = tab_76309[*py2++];
              *d2++ = clp[384+((y2 + c4)>>16)];         // Blue
              *d2++ = clp[384+((y2 - c2 - c3)>>16)];    // Green
              *d2++ = clp[384+((y2 + c1)>>16)];         // Red

          }
          d1 -= 9*width;
          d2 -= 9*width;
          py1+=    width;
          py2+=    width;
      }

}
```

5.參考文獻

1. Hao Pan's Homepage
   http://mri.beckman.uiuc.edu/pan/
2. DaubNET File Formats Collection BMP
   http://www.daubnet.com/formats/BMP.html#RasterData
3. BMP FILE
   http://home.kimo.com.tw/abc9250/BMP_FILE.htm
4. DIBs and Their Use
   http://support.microsoft.com/default.aspx?scid=kb;en-us;81498
5. MSDN
   http://www.msdn.com