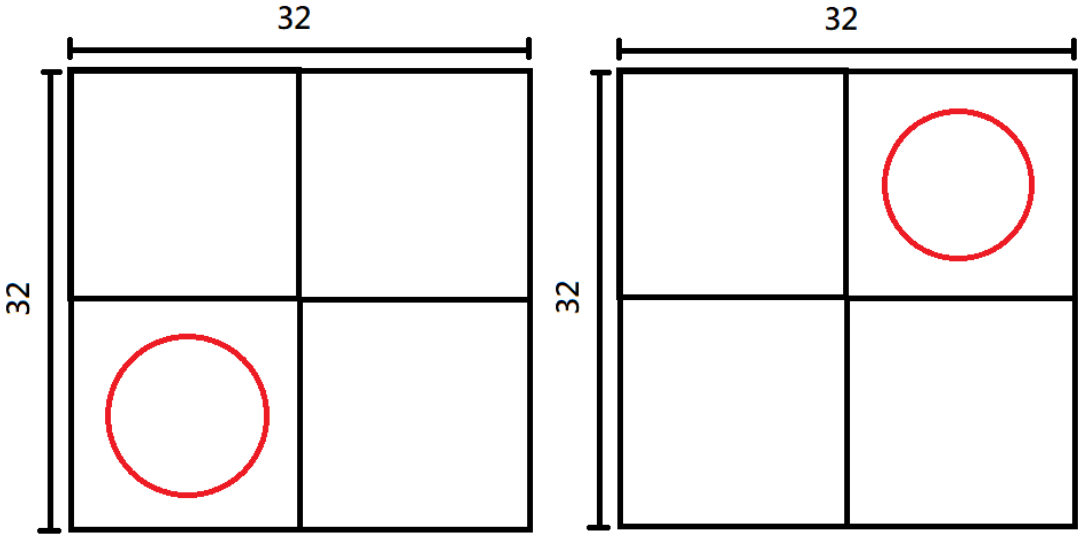
數位視訊技術

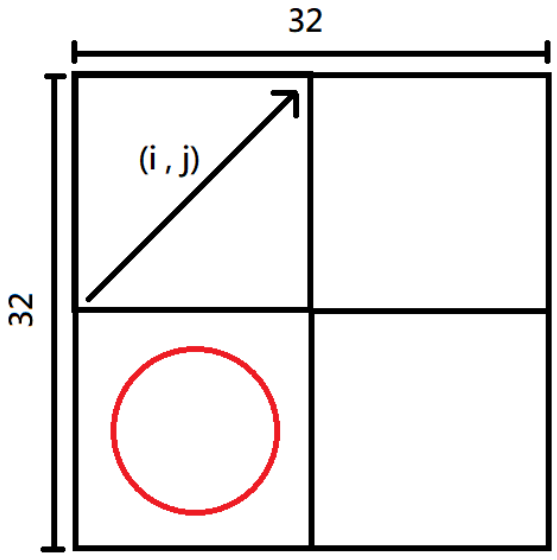
105323031 蔣圳嵐

原理

假設有一個圖片像素為 32x32，先將圖片切成若干個 block，而且每個 block 大小為 16x16，因此在這個案例內可以切成四個 block。再來把當前時間和過去時間的 frame 做比較，由圖一可以很明顯看出紅色圓圈從左下角跑到了右上角，因此就可以計算出他的 Motion Vector 也就是圖二的 (i , j) 向量。得到了 Motion Vector 即可使用當前圖像去預測未來的圖像。



圖一.左圖為前一時刻的 frame，右圖為這時刻的 frame



圖二.Motion Vector

而這個例子是當問題相當簡單的情況才能立即判斷紅色圈圈跑的方向，當一個圖片比較複雜的時候就需要仰賴一些數學的方法來判斷是往哪裡跑，有以下幾

種：

1. MAE(Mean Absolute Error)
2. MSE(Mean Squared Error)

而我選擇使用 MAE，公式如下：

$$\mathrm{MAE}(i,j)=\frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N}|x_t(k,l)-x_{t-1}(k+i,l+j)|$$

而要如何判斷是否是到那個位置，就是在搜尋範圍中每個 16x16 的 block 一個像素一個像素移動去尋找 MAE 的最小值，而我選定的搜尋範圍為[-16～+15]，最後找到的最小值 MAE 移動的 (i , j) 即為 Motion Vector。

程式碼和結果

```
img1 = imread("stefan_0_rgb.bmp");
img2 = imread("stefan_1_rgb.bmp");
img_out = img1(:,:,1);
MAE_min = 255*(16*16);

for k = 0:17
    for l = 0:21
        img2_in = double(img2(k*16+1:(k*16)+16,l*16+1:(l*16)+16,1));
        for i = -16 :15
            for j = -16:15
                if k*16+i+1>0 && l*16+j+1>0 && l*16+16+j<=352 && k*16+16+i<=288
                    img1_in = double(img1(k*16+i+1:(k*16)+16+i,l*16+j+1:(l*16)+16+j,1));
                    MAE = 1/(16*16)*sum(sum(abs(img2_in-img1_in)));
                    if MAE<MAE_min
                        MAE_min = MAE;
                        Vec_i = i;
                        Vec_j = j;
                    end
                end
            end
        end
        img_out(k*16+1:(k*16)+16,l*16+1:(l*16)+16,1) =...
            img_out(k*16+Vec_i+1:(k*16)+16+Vec_i,l*16+Vec_j+1:(l*16)+16+Vec_j,1);
        MAE_min = 255*(16*16);
    end
end

```

一開始先將圖片讀入矩陣 img1 和 img2，img1 為前一時刻 img2 為當前，之後再將預測的矩陣宣告為 img_out，因一開始都沒計算因此會和 img1 相同。
接下來就有四個 for 迴圈，第一個和第二個是在改變選定的 block 會分別從 0 到

17 及 0 到 21 是因為輸入圖片的像素為 288x352，每個 block 為 16x16，因此總共有(288/16)x(352/16) = 18x22 個 block。第三個和第四個 for 迴圈是在計算搜尋範圍中的 MAE 大小，當每個 Motion Vector 的 MAE 計算出來就去驗證是否為現在的最小值，如果是就取代最小的 MAE 並且將這個向量寫入 Vec_i 和 Vec_j。最後當搜尋範圍都找完即會得到一個 Motion Vector，再用這個 Motion Vector 將我們當前的 block 在我們要預測的圖片上移動 [Vec_i , Vec_j]，就可以得到一個預測的照片了。

左上為前一時刻的 frame，右上為當前時刻的 frame，下為預測的圖

結論

我試著將取得的 frame 拉比較遠，也就是相隔 4 個 frame 得到以下的結果：

突然恍然大悟，這不就是我們有時候看劇時，片源很差的時候會產生的一格一格模糊的情況嗎？因此這個現在才會不斷的推行 60 楨，因為這樣前後 frame 會更加連續，跑出來的樣子會更加的流暢，但當楨數到 60 的時候也會濃縮到我們的運算時間，所以硬體也勢必需要更新或是找到更加有效率的方法去尋找最小的 MAE。