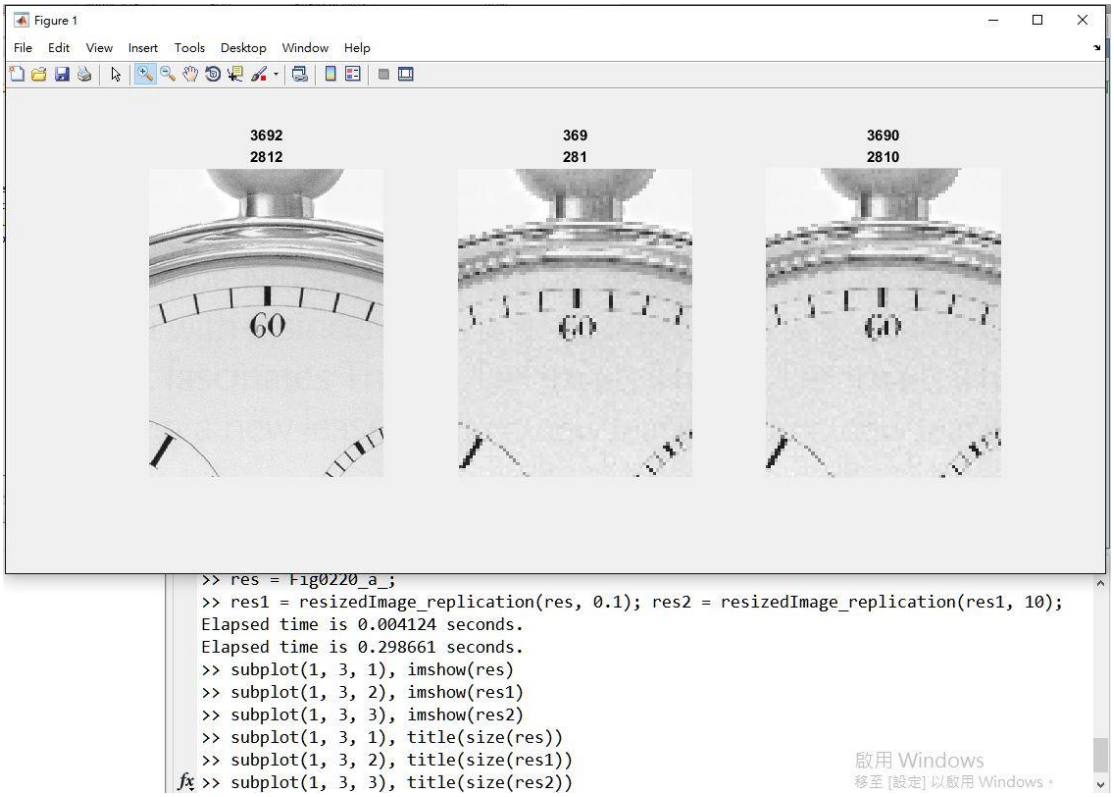
1. resizedImage_replication

(1) 作法說明：

使用的方法叫做 pixel replication，直接用新圖的新座標算出在原圖的對應點，然後把值 assign 到新圖。因為新圖 (1~n*f) 要對應到原圖 (1~n) ，因此我選擇使用取 ceiling 的方式， f(x) = ceil(x/f)，可以滿足對應關係，邊界的點也不用額外處理。

(2) 展示題目要求的結果並回答問題：

題目要求先 shrink 十倍再 zoom 十倍，比較他們的差異。以下是實際執行畫面：



觀察上面三張圖片，前兩張圖因為是 shrink，有些 pixel 會被

省略掉，因此細節部分會變模糊。而 zoom 不會省略掉 pixel ，因此細節部分看起來會和改變前一樣。

(3) 分析討論與觀察

a. Pre-allocation

因為程式執行時整個 matrix 是一個一個點 assign，matlab 會提醒我們要做 preallocate 來加速。實際測試後發現速度差很多，如果是 Fig0220(a) 的圖來跑放大兩倍，時間差了 30 倍以上，看輸出圖片的大小，有可能差到更多。因此 preallocate 是必要的。

b. 2D matrix(grayscale) and 3D matrix(RGB)

雖然題目圖片是 grayscale，但我有考慮到 3D matrix，使用 assign matrix(I, j, :) 的寫法，但是我發現這樣雖然 grayscale 和 RGB 都能執行，但回頭做 grayscale 的速度會比原本 assign matrix(I, j) 的寫法慢非常多，因此這裡就加個判斷，如果是 grayscale 就用 (I, j) ，如果是 RGB 就用 (I, j, :)

2. resizedImage_bilinear

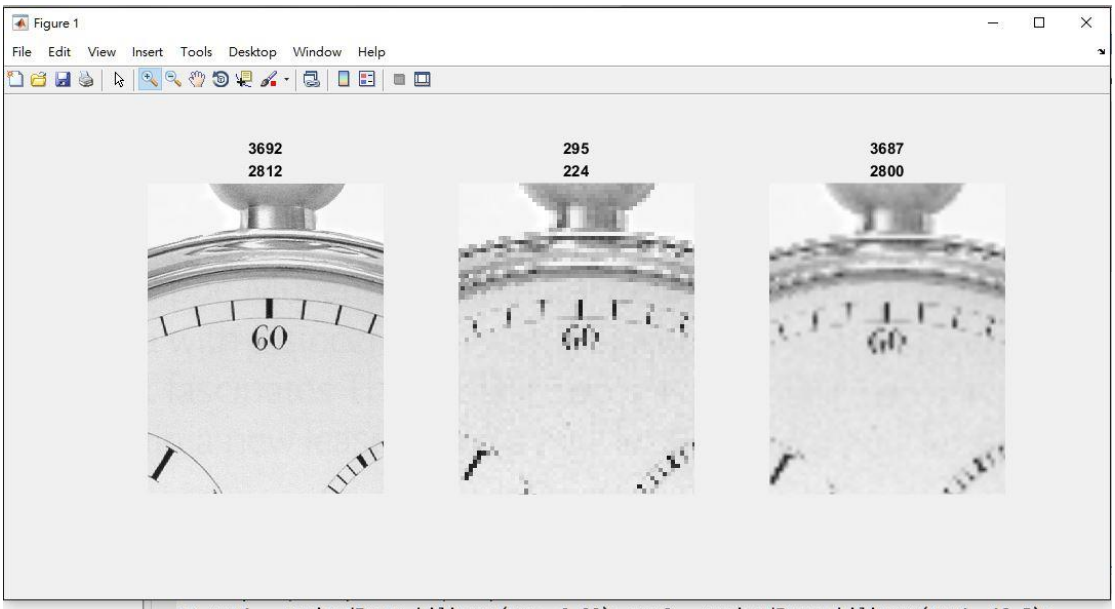
(1) 作法說明：

使用的方法叫做 bilinear interpolation，用新圖的新座標依

比例算出在原圖的對應點(座標可不必為整數)，然後用其周圍的四個點做內插法，算出較準確的值。因為 factor 是 pixel 的倍率而不是線段，我會先算出線段比例，再轉換成其在原圖的座標，然後就能套用講義上的內插法公式。

(2) 展示題目要求的結果並回答問題：

題目要求 dpi 1250 to dpi 100，換算縮小是 0.08 倍，放大是 12.5 倍，以下是執行結果：




```
>> res1 = resizedImage_bilinear(res, 0.08); res2 = resizedImage_bilinear(res1, 12.5);
Elapsed time is 0.008399 seconds.
Elapsed time is 0.979031 seconds.
>> subplot(1,3,1), imshow(res)
>> subplot(1,3,2), imshow(res1)
>> subplot(1,3,3), imshow(res2)
>> subplot(1,3,1), title(size(res))
>> subplot(1,3,2), title(size(res1))
>> subplot(1,3,3), title(size(res2))
```


可以看到在 zoom 部分， bilinear interpolation 細節處理會比 pixel replication 好，看起來會比較接近原圖。因為使用內插法，顏色的轉變會比較平滑，圖片看起來沒有一格一格的，比較有

漸層的感覺。

(3) 分析討論與觀察

a. Grayscale and RGB

這點和前面一樣，取值時特別判定，可以有效加速。

b. Fine tune

因為內插法需要除法和 ceil 和 floor 算相鄰的四個點，有可能會發生錯誤(ex: ceil(240.0000001) = 241)，因此需要做修正，取 ceil 前先減去一個小的數字，取 floor 前加上一個小的數字，防止溢出。