Lab01

107061123 孫元駿

Proj02-02: Reducing the Number of Intensity Levels in an Image

Explanation:

第一步，我將所有的值除 2^8，因為原始影像是 8 bits grayscale。這個步驟可以把值線性的限制在 $[0, 1)$。第二步，我將上一步的結果乘上 intensity level (k)。這一步可以線性放大所有值到 $[0, k)$。接著，對這個結果向下取整 (floor) ，這樣可以確保剩下的值會是 $\{ 0, 1, 2, \dots, k \}$。第三步，我會將上一步的結果乘上 $\frac{255}{k-1}$。$\frac{255}{k-1}$ 是每一個 step 的範圍，因此乘上它可以把原本$0 \sim (k-1)$ 的值對應到 0~255 之間。接著對值取整 (round)，確保每個值都是整數。結合上面所述，可以把公式合併成

$$resizedImage = round(floor((double(\frac{originalImage}{256}) * k) * (255/(k-1))$$

Proj02-03: Zooming and Shrinking Images by Pixel Replication

Explanation:

根據 Pixel Replication 的定義，在放大的時候會直接複製最接近點的值。

我將它分成兩個function，$resizeImage_replication(originalImage, scalingFactor)$和 $filledImage\_replication(a, b, c, d, newRows, newCols)$。$filledImage_replication$只負責處理 2×2 放大到 $newRows \times newCols$ 的處理。根據推導的結果，我們可以計算出每個 pixel 應該要填上哪一個值，再將整個矩陣回傳給 $resizeImage\_bilinear$，並且在 $resizeImage\_bilinear$ 中，最後再將回傳的矩陣貼上大的空矩陣，用這樣的方式可以減少運算的次數，也可以在未來利用平行運算的方式同時計算大量的資訊，有助於優化。

Conclusion:

根據觀察，因為示範的圖片顏色的對比度很高，銳利度也很高，因此用 Pixel Replication 的方式，並不會讓對比度改變很多，因此邊緣還是相對銳利的。但是可以明顯地發現縮小再法大的圖片在圓弧的部分會出現鋸齒，應該是因為縮小時太多資訊被刪除，加上用 Pixel Replication 會使影像出現一塊一塊相同顏色值的矩形。

Fig 1: Original Image.

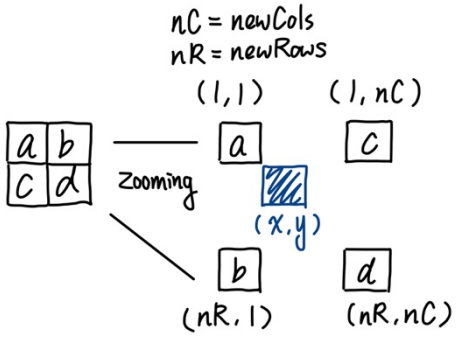
Fig2: Zooming and Shrinking by Pixel Replication

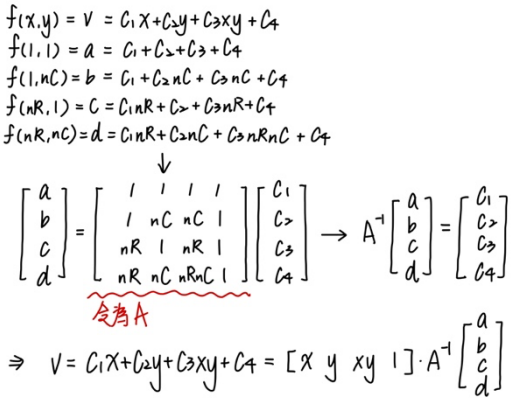
Proj02-04: Zooming and Shrinking Images by Bilinear Interpolation

Explanation:

根據 Bilinear Interpolation 的定義，在放大的時候會利用最接近的四個點線性的計算值。

我將它分成兩個function，$resizeImage_bilinear(originalImage, scalingFactor)$和 $filledImage\_bilinear(a, b, c, d, newRows, newCols)$。$filledImage_bilinear$只負責處理 2×2 放大到 $newRows \times newCols$ 的處理。





根據推導的結果，我們可以用 for loop 計算出每個 pixel 的值，再將整個矩陣回傳給 $resizeImage\_bilinear$，並且在 $resizeImage\_bilinear$ 中，將矩陣貼到正確的位置。因此，在我的程式中，我會以 2×2 的 window 分段去放大，這樣的做法可以有效的減少重新推導 $[C1 \quad C2 \quad C3 \quad C4]'$ 的次數。

Conclusion:

根據觀察，因為示範的圖片顏色的對比度很高，銳利度也很高，因此用 bilinear 的方式對這張圖片而言會過度的柔化邊緣，造成視覺上產生模糊的感覺。我們也可以從放大顯示每個 pixel 的方式，看出在很多地方都有漸層，但是原圖都是較為銳利的邊緣。

Fig 1: Original Image.

Fig 2: Zooming and Shrinking by Bilinear Interpolation.