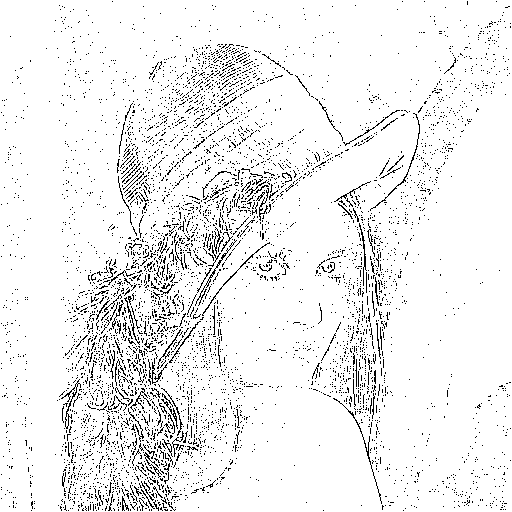
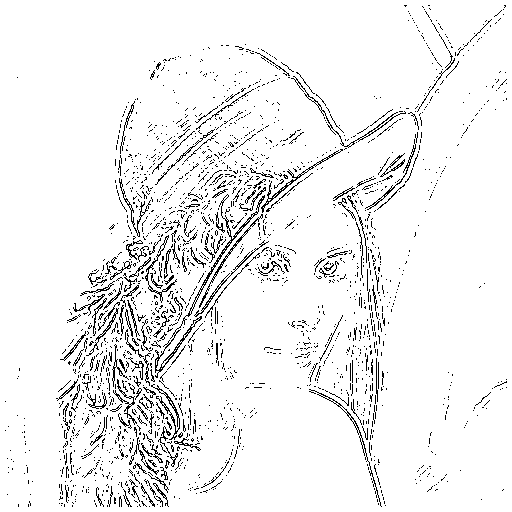
# CV Homework 10

R02922124 葉信良  
$ g++ -o hw10 hw10.cpp

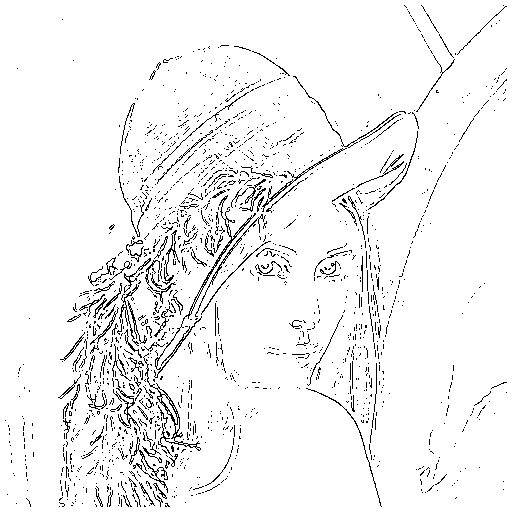
Part 1 Laplacian

我直接使用c++讀入bmp file lena.bmp，並將header 1078個字元讀出後直接輸出到output。接著將image的數值0~255存到unsigned char array BBMap中。   
然後依照投影片中的矩陣(我使用的是type 2，1/3那一個)，對每一個pixel做convolution總和算出tmp，接著把tmp存到tmpBMap矩陣對應的位置中；然後做zero crossing，如果一個點的周圍八個點有任一個點的tmp值小於-1 \* threhold的話就判斷是邊緣，設成0，其他則設成255。最後輸出OMap。  
而我比較了threhold 5~20的圖，覺得threhold=15時效果是比較好的。  
$ ./hw10 laplacian 15 圖片檔名是Laplacian\_15.bmp  


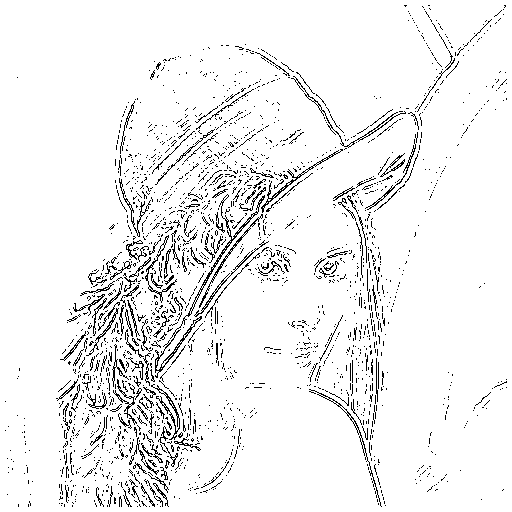
Part 2 minimum-variance Laplacian

我直接使用c++讀入bmp file lena.bmp，並將header 1078個字元讀出後直接輸出到output。接著將image的數值0~255存到unsigned char array BBMap中。   
然後依照投影片中的矩陣(正中間值是4/3那個矩陣)，對每一個pixel做convolution總和算出tmp，接著把tmp存到tmpBMap矩陣對應的位置中；然後做zero crossing，如果一個點的周圍八個點有任一個點的tmp值小於-1 \* threhold的話就判斷是邊緣，設成0，其他則設成255。最後輸出OMap。  
而我比較了threhold 5~20的圖，覺得threhold=14時效果是比較好的。  
$ ./hw10 mvlaplacian 14 圖片檔名是MVLaplacian\_14.bmp  


Part 3 Laplacian of Gaussian

我直接使用c++讀入bmp file lena.bmp，並將header 1078個字元讀出後直接輸出到output。接著將image的數值0~255存到unsigned char array BBMap中。   
然後依照投影片中的矩陣(11x11那個)，對每一個pixel做convolution總和算出tmp，接著把tmp存到tmpBMap矩陣對應的位置中；然後做zero crossing，如果一個點的周圍八個點有任一個點的tmp值小於-1 \* threhold的話就判斷是邊緣，設成0，其他則設成255。最後輸出OMap。  
而我比較了threhold 2000~4000的圖，覺得threhold=3000時效果是比較好的。  
$ ./hw10 laplagauss 3000 圖片檔名是LaplaGauss\_3000.bmp  


Part 4 Difference of Gaussian

我直接使用c++讀入bmp file lena.bmp，並將header 1078個字元讀出後直接輸出到output。接著將image的數值0~255存到unsigned char array BBMap中。   
然後依照自己計算出來的高斯的矩陣，對每一個pixel做convolution總和算出tmp，接著把tmp存到tmpBMap矩陣對應的位置中；然後做zero crossing，如果一個點的周圍八個點有任一個點的tmp值小於-1 \* threhold的話就判斷是邊緣，設成0，其他則設成255。最後輸出OMap。  
而我比較了threhold 0.5~10的圖，覺得threhold=5時效果是比較好的。  
$ ./hw10 diffgauss 5 圖片檔名是DiffGauss\_5.bmp  


🡪以上的source code、執行檔、result image皆附在壓縮檔中。