```
function y = im2jpeg_considerzeros(x, quality)

%im2jpeg Compress image x using JPEG

% reference: DIPUM

% LiuYin

% 2016-11-19


if ~ismatrix(x) || ~isa(x, 'uint8')

    error('The input x must be a UINT8 image.');

end


if nargin < 2

    quality = 1; % default quality

end


% normalization matrix

m = [16 11 10 16 24 40 51 61

    12 12 14 19 26 58 60 55

    14 13 16 24 40 57 69 56

    14 17 22 29 51 87 80 62

    18 22 37 56 68 109 103 77

    24 35 55 64 81 104 113 92

    49 64 78 87 103 121 120 101

    72 92 95 98 112 100 103 99] * quality;


% zig-zag order

order = [1 9 2 3 10 17 25 18 11 4 5 12 19 26 33 ...

       41 34 27 20 13 6 7 14 21 28 35 42 49 57 50 ...

       43 36 29 22 15 8 16 23 30 37 44 51 58 59 52 ...

       45 38 31 24 32 39 46 53 60 61 54 47 40 48 55 ...
```

```
       62 63 56 64];


[xm1, xn1] = size(x);

xm = ceil(xm1/8)*8;

xn = ceil(xn1/8)*8;

x = padarray(x,[xm-xm1 xn-xn1],0,'post');


x = double(x) - 128;

t = dctmtx(8);


% Compute DCTs of 8x8 blocks and quantize the coefficients.

fun_DCT = @(block_struct) t * block_struct.data * t';

y = blockproc(x, [8 8], fun_DCT);


fun_quantize = @(block_struct) round(block_struct.data ./ m);

y = blockproc(y, [8 8], fun_quantize);


y = im2col(y, [8 8], 'distinct');

xb = size(y, 2);


if 0% Show DC as image

   figure(10),clf,

   imshow(imresize(reshape(y(1,:),xm/8,xn/8),8,'nearest'),[])

   title('DC');

end


y = y(order, :); % reorder

eob = 1000;

r = zeros(numel(y) + size(y, 2), 1);
```

```
count = 0;
for j = 1:xb
    i = find(y(:,j), 1, 'last');    %每块中找出最后一个不为 0 的元素
        count = count + 1;              %计数加一,即当前的块序号

    %不为 0 的数为空
    if isempty(i)
        i = 0;
        r(count) = eob;                 %若不为 0 的数为空，则 i 置为 0，且用块结束标志 1000 去置位 r

    %存在最后一个不为 0 的元素
    else
        for k = 1:i

            %前面有连续的 0 元素
            if(y(k,j) == 0 && y(k+1,j)==0)
                w = k;
                while(y(w,j)==0)
                    w = w + 1;
                end
                r(count) = -1000;
                count = count + 1;
                r(count) = w - k;      %把每块中连续的 0 的个数算出，  这里是核心
                count = count + 1;
                k = w - 1;
            %前面没有连续（超过 1 个）0 的元素
            else
                r(count) = y(k,j);
                count = count + 1;
```

```
                end


            end
            r(count) = eob;                 % i 序号之后都是 0，再置 1000
        end
    end

r((count+1):end) = [];
clear y
y.original_size = uint16([xm1 xn1]);
y.size = uint16([xm xn]);
y.numblocks = uint16(xb);
y.quality = uint16(quality*100);

[symbols, prob] = prob4huffman(r);
dict = huffmandict(symbols,prob);
y.huffmanDict = dict;
hcode = huffmanenco(r,dict);
[y.huffmanCode,y.huffmanCodeLen] = huffmanDouble2Bin(hcode);
```

Published with MATLAB? R2015b