```
function x = jpeg2im_fixeddict(y,dict)  % 这个解码算法就是原始解码算法 jpeg2im.m
%jpeg2im Decode an IM2JPEG compressed image
% reference: DIPUM
% LiuYin
% 2016-11-19

% 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];

% 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];

rev = order;
for k = 1:length(order)
    rev(k) = find(order == k);
end

m = double(y.quality)/100*m;
```

```
xb = double(y.numblocks);

sz = double(y.size);

xn = sz(2);

xm = sz(1);


hcode = huffmanBin2Double(y.huffmanCode,y.huffmanCodeLen);

x = huffmandeco(hcode, dict);


eob = max(x(:));

z = zeros(64, xb);

k = 1;

for j = 1:xb

    for i = 1:64

        if x(k) == eob

            k = k+1;

            break

        else

            z(i, j) = x(k);

            k = k + 1;

        end

    end

end


z = z(rev, :);

x = col2im(z, [8 8], [xm xn], 'distinct');


fun_denormalize = @(block_struct) round(block_struct.data .* m);

x = blockproc(x, [8 8], fun_denormalize);
```

```
t = dctmtx(8);

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

x = blockproc(x, [8 8], fun_IDCT);

x = uint8(x+128);

x = x(1:y.original_size(1), 1:y.original_size(2));
```

Published with MATLAB? R2015b