```
function [ img_comp, tot_size ] = Haar_RecurCompress( img,
lim_vec, n )

% s = whos( 'img' );
% img_size = s.bytes;

lim = lim_vec(1);

img_r = img(:,:,1);
img_g = img(:,:,2);
img_b = img(:,:,3);

[img_hh_r, img_hl_r, img_lh_r, img_ll_r] =
Haar_Deconstruct( img_r );
[img_hh_g, img_hl_g, img_lh_g, img_ll_g] =
Haar_Deconstruct( img_g );
[img_hh_b, img_hl_b, img_lh_b, img_ll_b] =
Haar_Deconstruct( img_b );

%% hh %%
img_hh_r0 = zeros( size(img_hh_r) );
img_hh_r0( abs(img_hh_r) > lim ) = img_hh_r( abs(img_hh_r) >
lim );
num = sum( abs(img_hh_r(:)) > lim );

img_hh_g0 = zeros( size(img_hh_g) );
img_hh_g0( abs(img_hh_g) > lim ) = img_hh_g( abs(img_hh_g) >
lim );
num = sum( abs(img_hh_g(:)) > lim ) + num;

img_hh_b0 = zeros( size(img_hh_b) );
img_hh_b0( abs(img_hh_b) > lim ) = img_hh_b( abs(img_hh_b) >
lim );
num = sum( abs(img_hh_b(:)) > lim ) + num;

%% hl %%
img_hl_r0 = zeros( size(img_hl_r) );
img_hl_r0( abs(img_hl_r) > lim ) = img_hl_r( abs(img_hl_r) >
lim );
num = sum( abs(img_hl_r(:)) > lim ) + num;

img_hl_g0 = zeros( size(img_hl_g) );
img_hl_g0( abs(img_hl_g) > lim ) = img_hl_g( abs(img_hl_g) >
lim );
num = sum( abs(img_hl_g(:)) > lim ) + num;

img_hl_b0 = zeros( size(img_hl_b) );
img_hl_b0( abs(img_hl_b) > lim ) = img_hl_b( abs(img_hl_b) >
lim );
num = sum( abs(img_hl_b(:)) > lim ) + num;
```

```
%% lh %%
img_lh_r0 = zeros( size(img_lh_r) );
img_lh_r0( abs(img_lh_r) > lim ) = img_lh_r( abs(img_lh_r) >
lim );
num = sum( abs(img_lh_r(:)) > lim ) + num;

img_lh_g0 = zeros( size(img_lh_g) );
img_lh_g0( abs(img_lh_g) > lim ) = img_lh_g( abs(img_lh_g) >
lim );
num = sum( abs(img_lh_g(:)) > lim ) + num;

img_lh_b0 = zeros( size(img_lh_b) );
img_lh_b0( abs(img_lh_b) > lim ) = img_lh_b( abs(img_lh_b) >
lim );
num = sum( abs(img_lh_b(:)) > lim ) + num;

%% ll %%
img_ll_r0 = zeros( size(img_ll_r) );
img_ll_r0( abs(img_ll_r) > lim ) = img_ll_r( abs(img_ll_r) >
lim );
%num = sum( abs(img_ll_r(:)) > lim ) + num;
numll = sum( abs(img_ll_r(:)) > lim );

img_ll_g0 = zeros( size(img_ll_g) );
img_ll_g0( abs(img_ll_g) > lim ) = img_ll_g( abs(img_ll_g) >
lim );
%num = sum( abs(img_ll_g(:)) > lim ) + num;
numll = sum( abs(img_ll_g(:)) > lim ) + numll;

img_ll_b0 = zeros( size(img_ll_b) );
img_ll_b0( abs(img_ll_b) > lim ) = img_ll_b( abs(img_ll_b) >
lim );
%num = sum( abs(img_ll_b(:)) > lim ) + num;
numll = sum( abs(img_ll_b(:)) > lim )+ numll;

%% Data Type %%

img_hh_r0 = int8( img_hh_r0 );
img_hh_g0 = int8( img_hh_g0 );
img_hh_b0 = int8( img_hh_b0 );

img_hl_r0 = int8( img_hl_r0 );
img_hl_g0 = int8( img_hl_g0 );
img_hl_b0 = int8( img_hl_b0 );

img_lh_r0 = int8( img_lh_r0 );
img_lh_g0 = int8( img_lh_g0 );
img_lh_b0 = int8( img_lh_b0 );

img_ll_r0 = uint8( img_ll_r0 );
img_ll_g0 = uint8( img_ll_g0 );
img_ll_b0 = uint8( img_ll_b0 );
```

```
%% Size of non-ll %%

[w,h] = size( img_hh_r );
pix = 9*w*h;
size_nonll = num*8 + (pix - num);

if ( n == 1 )

    img_r = struct( 'hh', img_hh_r0, 'hl', img_hl_r0, 'lh',
img_lh_r0, 'll', img_ll_r0 );
    img_g = struct( 'hh', img_hh_g0, 'hl', img_hl_g0, 'lh',
img_lh_g0, 'll', img_ll_g0 );
    img_b = struct( 'hh', img_hh_b0, 'hl', img_hl_b0, 'lh',
img_lh_b0, 'll', img_ll_b0 );

    img_comp = struct( 'red', img_r, 'green', img_g, 'blue',
img_b );

    pixll = w*h*3;
    sizell = numll*8 + (pixll - numll);
    tot_size = size_nonll + sizell;
    return;

else

    img_ll = zeros(w,h,3);
    img_ll(:,:,1) = img_ll_r0;
    img_ll(:,:,2) = img_ll_g0;
    img_ll(:,:,3) = img_ll_b0;

    lim_vec1 = lim_vec(2:end);

    [img_comp1,sizell] = Haar_RecurCompress( img_ll, lim_vec1,
(n-1) );

    img_ll_r1 = img_comp1.red;
    img_ll_g1 = img_comp1.green;
    img_ll_b1 = img_comp1.blue;

    img_r = struct( 'hh', img_hh_r0, 'hl', img_hl_r0, 'lh',
img_lh_r0, 'll', img_ll_r1 );
    img_g = struct( 'hh', img_hh_g0, 'hl', img_hl_g0, 'lh',
img_lh_g0, 'll', img_ll_g1 );
    img_b = struct( 'hh', img_hh_b0, 'hl', img_hl_b0, 'lh',
img_lh_b0, 'll', img_ll_b1 );

    img_comp = struct( 'red', img_r, 'green', img_g, 'blue',
img_b );
    tot_size = sizell + size_nonll;

end
```

```
end
```