Another possibility is to use the [BWBOUNDARIES](http://www.mathworks.com/help/toolbox/images/ref/bwboundaries.html) function, which:

traces the exterior boundaries of objects, as well as boundaries of holes inside these objects

That information is contained in the fourth output A, an adjacency matrix that represents the parent-child-hole dependencies.

%# read binary image

bw = imread('SUvif.png');

%# find all boundaries

[B,L,N,A] = bwboundaries(bw, 8, 'holes');

%# exclude inner holes

[r,~] = find(A(:,N+1:end)); %# find inner boundaries that enclose stuff

[rr,~] = find(A(:,r)); %# stuff they enclose

idx = setdiff(1:numel(B), [r(:);rr(:)]); %# exclude both

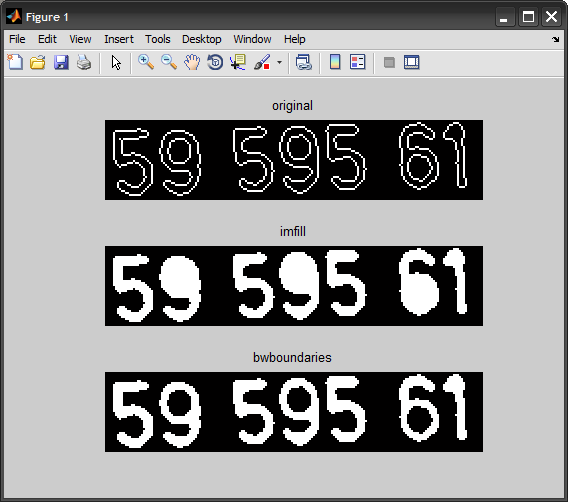
bw2 = ismember(L,idx); %# filled image

%# compare results

subplot(311), imshow(bw), title('original')

subplot(312), imshow( imfill(bw,'holes') ), title('imfill')

subplot(313), imshow(bw2), title('bwboundaries')



The problem is how to distinguish the holes from the digits. A possible ad hoc solution is filtering them by the area of the pixels inside.

function SolveSoProblem()

I = imread('http://i.stack.imgur.com/SUvif.png');

%Fill all the holes

F = imfill(I,'holes');

%Find all the small ones,and mark their edges in the image

bw = bwlabel(I);

rp = regionprops(bw,'FilledArea','PixelIdxList');

indexesOfHoles = [rp.FilledArea]<150;

pixelsNotToFill = vertcat(rp(indexesOfHoles).PixelIdxList);

F(pixelsNotToFill) = 0;

figure;imshow(F);

%Remove the inner area

bw1 = bwlabel(F,4);

rp = regionprops(bw1,'FilledArea','PixelIdxList');

indexesOfHoles1 = [rp.FilledArea]<150;

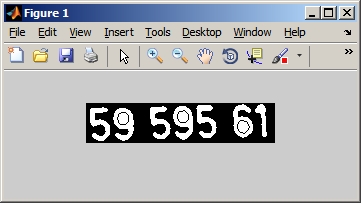
pixelListToRemove = vertcat(rp(indexesOfHoles1).PixelIdxList);

F(pixelListToRemove) = 0;

figure;imshow(F);

end

After **step(1)**:



After **step(2)**:

