```matlab
% make_saliency_maps_CogPsych.m
% adc's altered version of demonstration.m
% begun 2016-09-12
%
% NOTE: USES DOWNLOADED FUNCTION: freezeColors.m
% http://www.mathworks.com/matlabcentral/ [...]
%   fileexchange/7943-freezecolors---unfreezecolors


% Define directories

BASE_DIR = ['C:/Users/acate/Google Drive/teaching/' ...
    'CogPsych_4114_2016/in-class_assignments/' ...
    'assignment_3_image_saliency_maps/' ...
    ];

SCRIPT_DIR = [BASE_DIR 'code/'];

INPUT_DIR = [BASE_DIR 'submissions/'];

OUTPUT_DIR = [BASE_DIR 'output_images/'];

% Write files with no identifying student info. here, for sharing images
% with class.

ANON_OUTPUT_DIR = [BASE_DIR 'output_images_ANON/'];



% make a param. for which percentile of saliency map values to use when
% making cropped version of original image ("img_thresholded") below.  This
% is an INTEGER from 1 to 100.
%
% Orig. value in demonstration.m was 75

THRESH_PTILE = 95;


% From "demonstration.m" script distributed with the gvbs commands:
params = makeGBVSParams;
% could change params like this
params.contrastwidth = .11;



% Make all output images fit into an x-by-x box
outMaxDim = 400;

% Make struct array of directory contents
% First two entries will always be "." and ".."
f = dir(INPUT_DIR);

% Remove directories (including "." and "..") from list
f([f.isdir]) = [];

% Remove html files (which usually means that a student submitted a message
% in lieu of images).  Doing this is an alternate (and easier to code)
% method copmared to checking whether every file name has an image file
% extension.
f([...
    ~cellfun('isempty',regexp({f(:).name},'\.html$')) ...
    ]) = [];

% Students were instructed to submit two images: an original image and a
% marked version (which was usually a different size and file type: .gif).
% Marked versions were supposed to have the same file name but with a "2"
% appended before the "dot extension."
%
% adc cleaned up file names "by hand," removing whitespace characters and
% adding the "2" when needed.n


% Find indices of the originals, and assume that the index of each
% corresponding marked version is one greater.

origInds = find(cellfun('isempty',regexp({f(:).name},'2\..*$')));


% Make list of the student name strings that Canvas puts at the beginning
% of every file name.

personStrCell = cell(1,numel(f)); % empty cell array

for pp = 1:numel(f) % "pp" for "person"

    % Every file name downloaded from Canvas begins with "[student_name]_"
    personStrEndInd = regexp(f(pp).name,'^[a-z]*','end');
    personStrCell{pp} = f(pp).name(1:personStrEndInd);

end

% Make list of unique student (person) names
uniqueStrCell = unique(personStrCell);


% Make a list of randomly shuffled integers to use when
% writing anonymous file names; this avoids producing
% a list of names that preserves the alpha. order of student names.

% Take the index vector argout of matlab's "sort" for shuffled integers:
[sortY,anonInts] = sort(rand(numel(uniqueStrCell),1));


% "tic" and "toc" form a weird pair of matlab commands.  When "toc"
% executes, the time elapsed since "tic" is displayed on the matlab
% terminal.

tic

% MAIN LOOP
%
% For each student, load images, calculate saliency map, and draw nice
% figure.

for ii = 1:numel(uniqueStrCell)

    thisPerson = uniqueStrCell{ii};

    theseMatchInds = find(strcmp(thisPerson,personStrCell));

    thisOrigInd = intersect(origInds,theseMatchInds);
    thisMarkedInd = setdiff(theseMatchInds,origInds); % can be empty


    imOrig = imread([INPUT_DIR f(thisOrigInd).name]);

    imDims = size(imOrig); % can be 2 or 3 elements

    % include "min" to force this to be a scalar in case of equal h,w.
    biggerDim = min(find(imDims(1:2) == max(imDims(1:2))));

    scaleVec = [NaN, NaN];
    scaleVec(biggerDim) = outMaxDim;

    % resize image so that largest h,w dim equals outMaxDim,
    % while preserving aspect ratio
    imResized = imresize(imOrig,scaleVec);

    % this is how you call gbvs
    % leaving out params reset them to all default values (from
```

```matlab
    % algsrc/makeGBVSParams.m)
    imOut = gbvs(imResized);


    saliency_map = imOut.master_map_resized; % grayscale image

    if ( max(imResized(:)) > 2 ) imResized = double(imResized) / 255; end

    % Change to "<=" to EXCLUDE salient regions instead:
    imThresh = imResized .* repmat( ...
       saliency_map >= prctile(saliency_map(:),THRESH_PTILE) , ...
       [1, 1, size(imResized,3)] ...
       );

    % Now load the marked version, unless the next image file in alphabetical
    % order is also an "original."

    gifFlag = 0;

    if ~isempty(thisMarkedInd)
        thisMarkedImFN = [INPUT_DIR f(thisMarkedInd).name];
        if  isempty(regexpi(thisMarkedImFN,'\.gif$'))
            imMarked = imread([INPUT_DIR f(thisMarkedInd).name]);
            imMarkedResized = imresize(imMarked,scaleVec);
        else % is .gif
            [imMarked,imMarkedMap] = imread([INPUT_DIR f(thisMarkedInd).name]);
            [imMarkedResized,imMarkedResizedMap] = ...
              imresize(imMarked,imMarkedMap,scaleVec);
            gifFlag = 1;
        end
    else % Student didn't upload marked image
        imMarkedResized = zeros(size(imResized)); % image of zeros
    end

    f1 = figure(1);

    subplot(2,3,1);
    imshow(imResized);
    title('original image');

    subplot(2,3,2);
    imshow(saliency_map);
    freezeColors;
    title('Itti-Koch saliency map');

    subplot(2,3,4);
    if gifFlag
        imshow(imMarkedResized,imMarkedResizedMap);
    else
        imshow(imMarkedResized);
    end
    title('marked image');

    subplot(2,3,5);
    show_imgnmap(imResized,imOut);
    title('saliency map overlayed');

    subplot(2,3,6);
    imshow(imThresh);
    title(['most salient (' num2str(THRESH_PTILE) '%ile) parts']);


    % Make file names for the output images

    % For file names with student info. included:
    fnBase = f(thisOrigInd).name;
    fnBase = fnBase(1:end-4); % because all end in ".jpg"

    % Make sure "anonymous" file names do not sort into alpha. order of
    % student names; form will be "person1" e.g.
    anonBase = ['person' num2str(anonInts(ii))];

    % Save the output image variable (a struct) to a .mat file, one per image.
%     save([OUTPUT_DIR fnBase '_gvps.mat'],'imOut');
%     % Print the figure to an image file; do not include extension in file
%     % name arg., ".mat" is implied.
%     print(f1,'-dpng',[OUTPUT_DIR fnBase '_fig']);

    % Print anonymous file name versions
    save([ANON_OUTPUT_DIR anonBase '_gvps.mat'],'imOut');
    print(f1,'-dpng',[ANON_OUTPUT_DIR anonBase '_fig']);

    close(f1)

end


toc

```