```matlab
% Copyright 2012 Franziska Geringswald and Florian J. Baumgartner
% This program is free software: you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation, either version 3 of the License, or
% (at your option) any later version.
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
%
% This program can be used to validate a simulated central scotoma using a
% visual high-acuity, i.e. Landolt ring discrimination, task.
% If you use this program, please cite
% Geringswald, F., Baumgartner, F.J., & Pollmann, S. (2013).
% "A behavioral task for the validation of a
% gaze-contingent simulated scotoma", Behavior Research Methods.
% doi:10.3758/s13428-013-0321-6
%
% You should have received a copy of the GNU General Public License
% along with this program.  If not, see <http://www.gnu.org/licenses/>.

function runValidation
% Screen parameters have to be adapted to match the local setup. Setting
% the variable "tracking" to zero will run mouse simulation mode, otherwise
% the code is written to run with iViewX eye trackers based on the
% iViewXToolbox extension.
%
% The code was tested on Matlab version R2008b under Microsoft Windows XP SP3
using an iViewX Hi-Speed eye tracking
% system (SensoMotoric Instruments GmbH, Teltow, Germany) with a temporal
resolution of 240 Hz.

clear all
close all

try
    % for debugging only
    Screen('Preference', 'SkipSyncTests', 1);
    % Reseed the random-number generator for each experiment
    rand('state',sum(100*clock));
    % unify key names independently of operating system
    KbName('UnifyKeyNames')
    % Make sure we run on OpenGL Psychtoolbox
    AssertOpenGL;

    % tracking of gaze data; 0 = dummy mode, simulate gaze via mouse; 1 =
    % track gaze via eye tracker
    tracking = 1;

    % ----------------------------------
    % screen parameters
    % ----------------------------------
    scr=Screen('Screens');
    whichScreen=max(scr);
    param.screen.vDist = 60;                    % viewing distance
    param.screen.res   = get(0, 'ScreenSize');  % screen resolution in pixels
    param.screen.size  = [40 30];               % screen size in cm
    % ----------------------------------
    % design
    % ----------------------------------
    param.design.nblocks = 6;   % number of experimental blocks
    param.design.nstims = 8;    % number of stimulus positions on imaginary circle
    param.design.circ_size = 9; % size of imaginary circle in degree of visual
angle
    param.design.ngap = 4;      % number of gap positions
    param.design.nrep = 2;      % repetitions of each position x gap combination
per block

    % presentation times
    param.design.time.fix    = 1;   % fixation interval in sec
    param.design.time.iti    = .5;  % inter trial interval in sec
    param.design.time.search = 5;   % maximal duration of search display
presentation in sec
    param.design.time.vali   = 2;   % duration of each validation dot in sec

    % gap size of Landolt ring in pixels
    param.design.c_gap = 2;

    % instructions
    insEtSetup  = 'The eye tracker will now be set up. Please await further
instructions.';
    insSearch   = ['Find the "C" and indicate the direction of its opening with
the response keys.\n\n', 'Start by pressing any button'];
    insCaliTest = ['Fixate the dots one after another', '\n\n', 'Start by pressing
any button'];
    insFin      = 'Thank you for your participation';
    % response buttons (<, ^, >, v + q to exit)
    keycode = [KbName('LeftArrow') KbName('UpArrow') KbName('RightArrow')
KbName('DownArrow') KbName('q')]; % response key assignments

    % ----------------------------------
    % set up folders for logging and prompt basic experimental parameters
    % ----------------------------------
    p = what;
    resultsDir = [p.path '\results'];
    if ~isdir(resultsDir)
        mkdir(resultsDir)
    end

    % directory for storing gaze data, on remote machine controlling the
    % eye tracker, absolute path necessary
    eyeDir = 'D:\data\scoVal\';

    % prompt subject ID and create results folder
    param.subj.id = input('Please enter subject ID:', 's');
    param.subj.subjDir = [resultsDir '\' param.subj.id];
    if isdir(param.subj.subjDir)
        uID = datestr(now);
        uID = strrep(uID, ' ', '_');
        uID = strrep(uID, ':', '-');
        param.subj.subjDir = [resultsDir '\' param.subj.id '_' uID];
    end
    mkdir(param.subj.subjDir)
    disp(['Created subject directory ' param.subj.subjDir])
    % alternate scotoma presence across blocks; 1 = start with scotoma
    % in first block; 0 = start with scotoma in second block
    param.subj.ranscot = input('Scotoma block 1? 1 = yes 0 = no :', 's');
    tmpscot = param.subj.ranscot;
    % ------------------------------------
    % stimulus creation
    % ------------------------------------

    % calculate pixel sizes
    param.screen.px.cm     = param.screen.size./param.screen.res(3:4); % pixel
size in cm
    param.screen.px.dg     = (2*atan(param.screen.px.cm./
(2*param.screen.vDist))).*(180/pi); % pixel size in degree
    param.screen.px.arcmin = param.screen.px.dg*60; % pixel size in arcmin
    param.screen.px.perdg  = round(1./param.screen.px.dg(1));

    % define colors
    param.screen.cols.black = BlackIndex(whichScreen);
    param.screen.cols.white = WhiteIndex(whichScreen);
    param.screen.cols.gray  = round((param.screen.cols.white-
param.screen.cols.black)/2);

    % relation between visual acuity and eccentricity
    % Marmor, D.J., & Marmor, M.F. (2010). Simulating vision with and
    % without macular disease. Archives of Ophthalmology, 128,117-125.
    relvis(1,:) = [1 0.66666667 0.5 0.4 0.33333333 0.28571429 0.25 0.1 0.05 0.04
0.03333333]; % visual acuity
    relvis(2,:) = [0 .5 1 2 3 4 5 10 20 30 40]; % distance from fovea

    c_dia = param.design.c_gap*5; % diameter of landolt c

    findvis = 1/(param.design.c_gap*param.screen.px.arcmin(1)); % visual acuity
acquired to see gap
    findfov = interp1(relvis(1,:),relvis(2,:),findvis); % where on fovea can I
still see the gap?

    param.design.scotomaRad = findfov; % radius of absolute scotoma in degree
    param.design.scotomaCorr = .5; % gauss fading in degree
    param.design.scotomaSize =
round((param.design.scotomaRad+param.design.scotomaCorr)*param.screen.px.perdg*2);
% diameter of scotoma texture in pixel

    % create stimulus texture matrices
    [ring, sco, fix] = makeStims(param.screen.cols, param.screen.px.perdg,
param.design.scotomaSize);

    % stimulus coordinates
    [X,Y] = makeCircCoord(param.design.nstims, param.design.circ_size,
param.screen.px.perdg,[0 0 param.screen.res(3) param.screen.res(4)]);
    param.design.X=round(X);param.design.Y=round(Y);

    % randomize sequence
    param.design.sequence =
randomizer(param.design.nstims,param.design.ngap,param.design.nblocks,param.design
.nrep);
    % rotation angles of Landolt gap
    param.design.rot = [270 0 90 180];
    results.resp = zeros(1, size(param.design.sequence, 2),param.design.nblocks);
    results.rt   = results.resp;
    results.perf = results.resp;

    % feedback
    param.design.tone = [2000,500,22050,0.5]; % high pitch frequency, low pitch
frequency, sampling frequency, duration

     % open Window, AA set to 2 samples
    [w, rect] = Screen('OpenWindow', whichScreen, param.screen.cols.gray, [],[],2,
[],2);
    HideCursor;
    Screen('BlendFunction',w,'GL_SRC_ALPHA','GL_ONE_MINUS_SRC_ALPHA');

    % Do dummy calls to GetSecs, WaitSecs, KbCheck
    KbCheck;
    WaitSecs(0.1);
    GetSecs;

    % create stimulus textures
    % distractor and target
    t_ring(1) = Screen('MakeTexture', w, ring(:,:,1));
    t_ring(2) = Screen('MakeTexture', w, ring(:,:,2));
    % fixation
    t_fix = Screen('MakeTexture',w,fix);
    % scotoma
    t_scot = Screen('MakeTexture',w,sco);

    % points for eye tracker validation
    param.design.valipoints = calipoints(rect(3),rect(4),800,600);

    % eyetracker
    if tracking
        % present ET instruction
        ovals = abs(rect-50);
        oval_coord = [ovals(1), ovals(1); ovals(1), ovals(4); ovals(3), ovals(1);
ovals(3), ovals(4)];
        DrawFormattedText(w, insEtSetup, 'center', 'center',
param.screen.cols.white);
        Screen('FillOval', w, 255, recter(30,[oval_coord(:,1) oval_coord(:,2)],
rect));
        Screen('FillOval', w, 0, recter(4,oval_coord, rect));
        Screen('Flip', w);
        KbStrokeWait;
        % initialize eye tracker
        ivx = IViewXinitDefaults(w);
        iViewX('setchecklevel',ivx,3);
        ivx.absCalPos = param.design.valipoints;
        [success, ivx]=iViewX('openconnection', ivx);
        [success, ivx]=iViewX('datastreamingon', ivx);
        [success, ivx]=iViewX('clearbuffer', ivx);
        pnet(ivx.udp,'setreadtimeout',10^-6);
        [result, ivx] = iViewXCalibrate(ivx,1);
    end

    % present search instruction
    DrawFormattedText(w, insSearch, 'center', 'center', param.screen.cols.white);
    Screen('Flip', w);
    KbStrokeWait;

    % create logfiles
    % search experiment
    resFileName = [param.subj.subjDir '\' param.subj.id];
    fid = fopen(resFileName, 'a');
    fprintf(fid, '%s\n', [param.subj.id]);
    fprintf(fid, '%s\n', [datestr(now)]);
    fprintf(fid, '%s\n', ['blk trl cond targ_pos gap resp rt targ_X targ_Y']);

    % save information about validation points
    valiFileName = [param.subj.subjDir '\' param.subj.id '_valInfo'];
    vid = fopen(valiFileName, 'a');
    fprintf(vid, '%s\n', [param.subj.id]);
    fprintf(vid, '%s\n', [datestr(now)]);
    fprintf(vid, '%s\n', ['blk trl X Y']);

    % save .mat file containing experiment parameters
    save([resFileName '_setup'], 'param');

    for k=1:param.design.nblocks

        if rem(tmpscot,2)
            scotoma = 1;
        else
            scotoma = 0;
        end

        % present number of current block
        dg = ['Block ', num2str(k), ' of ', num2str(param.design.nblocks)];
        DrawFormattedText(w, dg, 'center', 'center', param.screen.cols.white)
        Screen('Flip', w);
        KbStrokeWait;

        % Clear screen to background color
        Screen('Flip', w);

        % Wait a second before starting trial
        WaitSecs(1.000);

        if tracking
            iViewX('startrecording', ivx);
        end

        % trial loop
        for i=1:size(param.design.sequence,2)
            cor = [0, 0];
            % fixation
            FIX = GetSecs;
            if tracking
                iViewX('message',ivx,['fixation_on' ])
            end
            while GetSecs-FIX <= param.design.time.fix;
                Screen('DrawTexture', w, t_fix, [],
recter(param.screen.px.perdg*9, [rect(3)/2, rect(4)/2]),[],1);
                if scotoma
                    if tracking
```

```
                        % latest gaze sample available
                        data = iViewX('receivelast',ivx);
                        if data ~=-1 & strcmp(data(4:6),'SPL')
                            J = strread(data,'%s');
                            cor = [str2double(J{3}),str2double(J{4})];
                        end
                    else
                        [mouse_x, mouse_y] = GetMouse;
                        cor = [mouse_x, mouse_y];
                    end
                    Screen('DrawTexture',w,t_scot,
[],recter(param.design.scotomaSize, [cor(1),cor(2)],rect));
                end

                Screen('Flip', w);
            end

            keyisdown = 0;
            keyFlag   = 0;
            secs = 0;

            % while loop to show stimulus until subjects response or until
            % "duration" seconds elapsed.
            initTime = GetSecs;

            % distractor positions
            distr = setdiff([1:8],param.design.sequence(1,i,k));
            if tracking
                iViewX('message',ivx,['display_on' ])
            end

            while 1
                Screen('DrawTextures',w,t_ring(1),[],recter(c_dia,
[param.design.X(distr),param.design.Y(distr)],rect),[],1);
                Screen('DrawTexture', w, t_ring(2), [], recter(c_dia,
[param.design.X(param.design.sequence(1,i,k)),param.design.Y(param.design.sequence
(1,i,k))]),[param.design.rot(param.design.sequence(2,i,k))],1);
                if scotoma
                    if tracking
                        % latest gaze sample available
                        data = iViewX('receivelast',ivx);
                        if data ~=-1 & strcmp(data(4:6),'SPL')
                            J = strread(data,'%s');
                            cor = [str2double(J{3}),str2double(J{4})];
                        end
                    else
                        [mouse_x, mouse_y] = GetMouse;
                        cor = [mouse_x, mouse_y];
                    end
                    Screen('DrawTexture',w,t_scot,
[],recter(param.design.scotomaSize, [cor(1),cor(2)],rect));
                end

                Screen('Flip', w);


                [keyisdown, secs, key]=KbCheck;
                if ismember(find(key(1:end)), keycode)
```

```
                    key=KbName(key);
                    if strcmp(key,'q')
                        fclose('all');
                        Screen('CloseAll');
                        if tracking
                            iViewX('stoprecording', ivx);
                            [success, ivx]=iViewX('datastreamingoff', ivx);
                            [success, ivx]=iViewX('closeconnection', ivx);
                        end
                        return
                    end
                    % ------------------------
                    results.rt(i,k) = (secs-initTime)*1000;
                    if strcmp(key, 'LeftArrow')   results.resp(i,k) = 1;
                    elseif strcmp(key, 'UpArrow') results.resp(i,k) = 2;
                    elseif strcmp(key, 'RightArrow') results.resp(i,k) = 3;
                    elseif strcmp(key, 'DownArrow') results.resp(i,k) = 4;
                    end;
                    fprintf(fid, '%d %d %d %d %d %d %d %d %d\n', [k i scotoma
param.design.sequence(1,i,k) param.design.sequence(2,i,k) results.resp(i,k)
round(results.rt(i,k)) param.design.X(param.design.sequence(1,i,k))
param.design.Y(param.design.sequence(1,i,k))]);
                    if results.resp(i,k)==param.design.sequence(2,i,k)
                        results.perf(i,k) = 1; tone =
makebeep(param.design.tone(1),param.design.tone(4));
                    else
                        results.perf(i,k) = 0; tone =
makebeep(param.design.tone(2),param.design.tone(4),param.design.tone(3));
                    end
                    keyFlag = 1;
                    if tracking
                        iViewX('message',ivx,['response' ])
                    end
                end
                if GetSecs-initTime > param.design.time.search |  keyFlag == 1
                    break
                end
            end

            if keyFlag==0
                results.resp(i,k) = 100; results.rt(i,k)=-1;
                tone =
makebeep(param.design.tone(2),param.design.tone(4),param.design.tone(3));
                fprintf(fid, '%d %d %d %d %d %d %d %d %d\n', [k i scotoma
param.design.sequence(1,i,k) param.design.sequence(2,i,k) results.resp(i,k)
round(results.rt(i,k)) param.design.X(param.design.sequence(1,i,k))
param.design.Y(param.design.sequence(1,i,k))]);
                if tracking
                    iViewX('message',ivx,['response' ])
                end
            end

            % play feedback sound
            sound(tone,param.design.tone(3));

            %ITI
            ITI = GetSecs;
            while GetSecs-ITI <= param.design.time.iti;
                Screen('Flip', w);
            end

            if tracking
                sets = i;
                iViewX('incrementsetnumber', ivx, num2str(sets));
            end
        end

        if tracking
            iViewX('stoprecording', ivx);
            iViewX('datafile', ivx, [eyeDir, param.subj.id, '_lan_', num2str(k)]);
        end

        % ET validation after each block
        DrawFormattedText(w, insCaliTest, 'center', 'center',
param.screen.cols.white);
        Screen('Flip', w);
        KbStrokeWait;
        if tracking
            iViewX('startrecording', ivx);
        end
        for i = 1:length(param.design.valipoints)
            valiset=i;
            VALI = GetSecs;
            while GetSecs-VALI <= param.design.time.vali
                Screen('FillOval',w,[0,0,0], recter(10,
[param.design.valipoints(i,1) param.design.valipoints(i,2)], rect));
                if ~tracking
                    [mouse_x, mouse_y] = GetMouse;
                    Screen('FillOval',w,[255,0,0], recter(10, [mouse_x mouse_y],
rect));
                end
                Screen('Flip',w);

                [a,b,c]=KbCheck;
                if a
                    if strcmp(KbName(c), 'ESCAPE')
                        break
                    end
                end
            end
            fprintf(vid, '%d %d %d %d\n', [k i param.design.valipoints(i,1)
param.design.valipoints(i,2)]);
            if tracking
                iViewX('incrementsetnumber', ivx, num2str(valiset));
            end
        end
        if tracking
            iViewX('stoprecording', ivx);
            iViewX('datafile', ivx, [eyeDir, param.subj.id,  '_val_',
num2str(k)]);
        end
        tmpscot=tmpscot+1;
    end
    % Done. Show cursor and close window.
    DrawFormattedText(w, insFin, 'center', 'center', param.screen.cols.white);
    Screen('Flip', w);
    fclose('all');

    % save .mat file containing experiment results
    save([resFileName '_results'], 'results');

    if tracking
        [success, ivx]=iViewX('datastreamingoff', ivx);
        [success, ivx]=iViewX('closeconnection', ivx);
    end

    KbWait;
    ShowCursor;
    Screen('CloseAll');

catch
   ShowCursor;
   fclose('all');
   if tracking
       iViewX('stoprecording', ivx);
       [success, ivx]=iViewX('datastreamingoff', ivx);
       [success, ivx]=iViewX('closeconnection', ivx);
   end
   Screen('CloseAll');
   psychrethrow(psychlasterror);
end
end

function M = gauss(x,s,r)
% M = gauss(x,s,r)
% Creating 2-dim standardized Gaussian distribution
% x: int length of the window
% s: float standard deviation
% r: int cut-off value for creating a plateau
    if nargin == 2
        r = 0;
    elseif nargin == 1
        r = 0;
        s = 1;
    elseif nargin == 0
        r = 0;
        s = 1;
        x = 100;
    end
    G = repmat(exp(-(linspace(-3,3,x)).^2./(2*s^2)),x,1);
    M = G.*G';
    M = M-min(min(M));
    M = M/max(max(M));

    k = max(M(x/2+r/2,:));
    M = M/k;
    M(M>1) = 1;
    M = M *255;
end

function rect = recter(siz, coord, Rect, shift)
```

```
% rect = recter(siz, coord, Rect, shift)
% Computes PTB coordinate points of a rectangle
% siz : [int,int]  size of the rectangle (if len(siz)==1 square)
% coord: [float,float]  center coordinates of the rectangle (if coord<1 relative
to Rect)
% Rect: [int,int,int,int]  reference frame if coord<1
% shift: [int,int] shift of the center of the rectangle
    if nargin == 3
        shift = [0,0];
    elseif nargin == 2
        Rect = get(0, 'ScreenSize');
        shift = [0,0];
    elseif nargin == 1
        coord = [0.5,0.5];
        Rect = get(0, 'ScreenSize');
        shift = [0,0];
    end

    if size(siz,2) == 1
        siz = [siz,siz];
    end

    if coord(1)>=1
        rectx1 = -siz(:,1)/2+shift(:,1)+coord(:,1)+1;
        recty1 = -siz(:,2)/2+shift(:,2)+coord(:,2)+1;
        rectx2 = siz(:,1)/2+shift(:,1)+coord(:,1);
        recty2 = siz(:,2)/2+shift(:,2)+coord(:,2);
    else
        rectx1 = -siz(:,1)/2+shift(:,1)+Rect(3)*coord(:,1)+1;
        recty1 = -siz(:,2)/2+shift(:,2)+Rect(4)*coord(:,2)+1;
        rectx2 = siz(:,1)/2+shift(:,1)+Rect(3)*coord(:,1);
        recty2 = siz(:,2)/2+shift(:,2)+Rect(4)*coord(:,2);
    end
    rect = [rectx1,recty1,rectx2,recty2];

    if size(rect,1) > 1
        rect = rect';
    end
end

function [out] = makeCirc(stimSize)
% [out] = makeCirc(stimSize)
% creates a circular disk of size 'stimSize' which is used for stimulus
% generation
    [x,y] = meshgrid(linspace(-stimSize, stimSize, 2*stimSize), linspace(-
stimSize, stimSize, 2*stimSize));
    d     = sqrt(x.^2+y.^2);
    out   = d<=stimSize;
end

function [ring, sco, fix] = makeStims(cols, pixel, scotomaSize)
% [ring, sco, fix] = makeStims(cols, pixel, scotomaSize)
% creates stimulus texture matrices for Landolt ring and closed circles
% (upscaled to s), the simulated scotoma patch of the size 'scotomaSize' in
% degree and a fixation stimulus.
    s = 100;
    out_tmp = makeCirc(s/2*5);
    in_tmp  = makeCirc(s/2*3);
    dist = zeros(size(out_tmp));
    in  = zeros(size(in_tmp));

    % distractor
    dist(out_tmp==0)= cols.gray;
    dist(out_tmp==1)= cols.black;
    in(in_tmp==0)  = cols.black;
    in(in_tmp==1)  = cols.gray;
    dist(s+1:s*4, s+1:s*4)=in;
    %target
    tar = dist;
    st = s*5/2-s/2+1;
    ed = s*5/2+s/2;
    tar(1:s*5/2,st:ed)=cols.gray; % gap

    ring(:,:,1)=dist;
    ring(:,:,2)=tar;

    % scotoma
    sco = ones(scotomaSize,scotomaSize,4)*cols.gray;
    if rem((scotomaSize*2-pixel),2)
        sco(:,:,4) = gauss(scotomaSize,1,scotomaSize-(pixel+1));
    else
        sco(:,:,4) = gauss(scotomaSize,1,scotomaSize-pixel);
    end

    % fixation
    rad   = round(10*pixel);
    radIn = round(.3*rad);
    radii = linspace(radIn, rad, 3);
    lwd = 5;

    [x,y] = meshgrid(linspace(-rad,rad, 2*rad), linspace(-rad,rad,2*rad));
    d     = sqrt(x.^2+y.^2);

    for i = 1:3
        ann(:,:,i) = (abs(d<=radii(i))&~abs(d<=radii(i)-lwd));
    end;

    fix = sum(ann, 3);
    fix(fix==1)=cols.white;
    fix(fix==0)=cols.gray;
    cr = eye(size(fix,1));
    fix(cr==1)=cols.white;
    fix(flipud(cr)==1)=cols.white;
end

function [X,Y] = makeCircCoord(nstims, circ_size, pixel,rect)
% [X,Y] = coords(nstims, circ_size, pixel, rect)
%
% Returns X and Y coordinates of n items ('nstims') on n concentric circles
(length of 'circ_size') with radii
% of n times one degree of visual angle ('pixel') in a given area ('rect')
%
% n_stims = number of stimuli, vector if more than one circle
% circ_size(1:2) = size of inner and outer circle in degree
% pixel = number of pixel per degree
% rect = size of window
%
% nstims = [8 16];
% circ_size = [5 8];
% pixel = pixel;
% rect=rect;

    if isempty(rect)
        a = get(0, 'ScreenSize');
        rect = [0 0 a(3) a(4)];
    end

    n_circ=max(size(nstims));

    if max(circ_size)*pixel >= rect(4)/2
        disp('cut circle?');
        cut = input('1=cut, 0=shrink: ');

        if cut == 1
            radius(1:n_circ) = linspace(min(circ_size)*pixel,
max(circ_size)*pixel, n_circ);
        else
            siz = input('stimsize [px]: ');
            if n_circ == 1
                radius = rect(4)/2-siz;
            else
                radius(1:n_circ) = linspace(min(circ_size)*pixel, rect(4)/2-siz,
n_circ);
            end
        end
    else
        radius(1:n_circ) = linspace(min(circ_size)*pixel, max(circ_size)*pixel,
n_circ);
    end

    % stimulus coordinates on cicle
    if max(size(nstims))==1
        theta = linspace(2*pi/nstims, 2*pi,nstims);
        [X,Y] = pol2cart(repmat(theta',1,n_circ), repmat(radius,nstims,1));
        % translate to screen coords
        %           2
        %        3     1
        %      4         8
        %        5     7
        %           6
        X = X+rect(3)/2;
        Y = (Y-rect(4)/2)*-1;
    else
        for i=1:n_circ
            theta{i} = linspace(2*pi/nstims(i), 2*pi, nstims(i));
            [X{i},Y{i}] = pol2cart(theta{i}, radius(i));
            X{i} = X{i}+rect(3)/2;
            Y{i} = (Y{i}-rect(4)/2)*-1;
        end
    end
end

function [sequence] = randomizer(nstims,ngap,nblocks, nrep)
```

```
% [sequence] = randomizer(nstims,ngap,nblocks, nrep)
% Returns a three dimensional matrix containing target stimulus parameters. Each
% column contains pairs of target position ('nstims', first row) and gap
orientation
% ('ngap', second row), the unique combinations being repeated 'nrep' times and
% shuffled for each experimental block ('nblocks', third dimension).
  cond = [repmat(sort(repmat(1:nstims,1,ngap)),1,nrep);
repmat(repmat(1:ngap,1,nstims),1,nrep)];
  sequence = zeros(size(cond,1), size(cond,2),nblocks);

  for i=1:nblocks
    ran = randperm(size(cond,2));
    sequence(:,:,i) = cond(:,ran);
  end
end

function C = calipoints(resx,resy,picx,picy)
% C = calipoints(resx,resy,picx,picy)
% List of 13 equally distributed fixation point
% resx,resy : int, int x and y dimension of the reference frame
% picx,picy : int, int max. x and y dispersion from the center of the
% reference frame
    X = resx/2;
    Y = resy/2;

    x = picx/2;
    y = picy/2;

    C = [
    X, Y;...
    X-x, Y-y;...
    X+x, Y-y;...
    X-x, Y+y;...
    X+x, Y+y;...
    X-x, Y;...
    X, Y-y;...
    X+x, Y;...
    X, Y+y;...
    X-x/2, Y-y/2;...
    X+x/2, Y-y/2;...
    X-x/2, Y+y/2;...
    X+x/2, Y+y/2];
end
```