```
% This script is a combination of Simulation_7_2.m and Speckles_Sim_2
% features from 8_1 not included
% 8_2: Speckle cross section "cigars"
%Speckle_Sim_2 Speckle simulation with spherical waves

clc; clear all; close all;
load('waveOrigin_2018-07-18.mat')


%% GLOBAL: parameters

lambda = 0.00053;   %mm
ff = 50;            %mm

surfaceVariance = 0.00003;  %mm
zPlanes = 10;       %mm

apertureSize = 0.5; %mm
Dp_slm = 0.008;     %mm

%% SPECKLES: parameters
res = 1024;
dx = lambda*ff/(Dp_slm*res);       %pixel size (theses are chosen so that the sampling↙
of the SLM and the fft overlap)
du_4f = 1/(res*dx);    %pixel size in fourier domain in 1/mm for propagation of the↙
right plane

%% SPECKLES: Wavefield Calculation
NN = 128; %Number of Waves

%waveOriginX = (round(rand(NN,1)*(res+1)-0.5)-res/2)*dx;
%waveOriginY = (round(rand(NN,1)*(res+1)-0.5)-res/2)*dx;

%dz = randn(NN,1)*surfaceVariance;

[screenX, screenY] = meshgrid(dx*(-res/2+1:res/2), dx*(-res/2+1:res/2));

waveField = zeros(res);
for ii = 1:NN
    waveField = waveField + exp(1i*2*pi/lambda*sqrt((zPlanes+dz(ii)).^2+↙
(screenX+waveOriginX(ii)).^2 + (screenY+waveOriginY(ii)).^2));
    ii
end

imgPlaneWavefield = waveField;

%%↙
=========================================================================================↙
=================
%% SPICE: parameters

res = length(imgPlaneWavefield);

du_4f = lambda * ff / (dx * res);
```

```
%% initialization
fourierAperture = zeros(res);
waveField = imgPlaneWavefield;
u_zCrossSection = NaN(res);

%% SPICE: first lens
%DFT_il = ifftshift(fft2(double(imread('einstein.bmp'))));
FwaveField = fftshift(fft2(double(waveField)));

%% SPICE: aperture
fourierAperture(ceil(res/2),ceil(res/2)) = 1;
fourierAperture = (bwdist(fourierAperture) <= apertureSize/2/du_4f);       %aperture in fourier domain

FapertureWaveField = FwaveField .* fftshift(fourierAperture);

%% SPICE: SLM propagatiom

[uu,vv] = meshgrid(-res/2+1:res/2, -res/2+1:res/2);
uu = du_4f*uu;
vv = du_4f*vv;

correlationCoeff = zeros(res+1,1);

%% calculation of reference image for zSLM = zPlanes
transferFunction = ((exp(-1i * 2*pi/lambda * zPlanes * sqrt( 1 -  (uu.^2 + vv.^2) / ff^2))));
if du_4f == Dp_slm
    U_rPlanes = fftshift(transferFunction).*FapertureWaveField;
else
    display('Error: du_4f must be equal to Dp_slm')
    return
end
u_zPlanes = ifft2(U_rPlanes);

%% scan through the wavefield
zSLM = 500;

for ll = 1:length(zSLM) %scan through the entire wavefield
    transferFunction = ((exp(-1i * 2*pi/lambda * zSLM(ll) * sqrt( 1 -  (uu.^2 + vv.^2) / ff^2))));
    if du_4f == Dp_slm
        U_r = fftshift(transferFunction).*FapertureWaveField;
    else
        display('Error: du_4f must be equal to Dp_slm')
        return
    end
    u_z = ifft2(U_r);
    u_zSlice = u_z(:,res/2);
    u_zCrossSection(:,ll) = u_zSlice;
    correlationCoeff(ll) = corr2(abs(u_z),abs(u_zPlanes));    %correlation with zSLM=zPlanes image
    ll
end
```

```
%figure, imshow(angle(u_z),[])
%figure, imshow(abs(u_z),[])
%figure, imshow(abs(u_zCrossSection),[])
histogram = imhist(abs(u_z).^2);

lorentzFit = fittype('a*z_max^2/((zSLM-c)^2+z_max^2)', 'dependent',↙
{'correlation_coefficient'},'independent',{'zSLM'},'coefficients',{'a','z_max','c'});
myFit = fit(zSLM.', correlationCoeff, lorentzFit)
%figure, plot(myFit,zSLM,correlationCoeff)

%figure, plot(histogram)
```