INTRODUCTION

For this visualization we will start with a simple binary image of circuit lines and use noise mixtures and pixel colors to create a realistic 3D model of a circuit board.

BEGINNING OF TASK

Please follow the tutorial and reproduce the figures with your own code.

The first step is to input a binary image of the circuit and establish transformation coefficients between lateral samples, image intensity and world coordinates. We will be using this circuit pattern as the footprint for a 3D topology.

```matlab
% noise mixtures
clear all;
dx=50/72; % (microns per pixel) physical distance between pixels
dy=dx;
dz=50; % (microns per pixel intensity) physical distance for image intensity of 1
% input mask image
A_bmp=double(imread('SimpleCircuit.bmp')); % load example.bmp image
Ar=A_bmp(:,:,1);
Ag=A_bmp(:,:,2);
Ab=A_bmp(:,:,3);
Amask=(Ar+Ag+Ab)/3;
[My Nx]=size(Amask);
% figure(1)
imagesc(Amask)
colormap gray;
axis image;
title('Circuit feature map')
print -djpeg fig1
```

Figure 1: Original image.
Some images may not be fully binarized so a binarization step is implemented.

\[
\text{Cmask} = \text{binarize}(\text{Amask});
\]

Figure 2: Binarized image where white has value 1 and black is value 0.

The white values are the “foreground” conductive material on top of the “background” substrate. The image intensity will represent the Z coordinate of the surface.

3-D SURFACE MODEL

The first step in creating a realistic surface is to smooth the edges. We will use a Gaussian shaped filter such that

\[
\begin{align*}
\text{sigma} &= 4; \\
\text{x} &= 1: \text{Nx}; \\
\text{x} &= \text{x} - \text{Nx}/2; \\
\text{y} &= 1: \text{My}; \\
\text{y} &= \text{y} - \text{My}/2; \\
\text{hgauss} &= \text{gaperture}(\text{sigma}, \text{sigma}, \text{x}, \text{y}); \\
\text{hgauss} &= \text{fftshift}(\text{hgauss}); \quad \% \text{shift to put origin at the corners}
\end{align*}
\]

Figure 3: Gaussian filter setup with origin in upper left corner.

It is very hard to see but the spatial origin of the filter in Fig. 3 is in the upper left and the Gaussian shape wraps around to the other corners. When circularly convolved with the input
mask, the result will not shift spatially. Later we will show the filters with the origin in the center of the images so it is easier to visualize. The upsampled image is convolved with the filter such that

```matlab
Dmask = real(ifft2(fft2(Cmask).*fft2(hgauss)));
Dmask = Dmask-min(min(Dmask));
Dmask = Dmask./max(max(Dmask));
```

The result will alter the background substrate depth so to keep the model realistic we extend the foreground to include these regions using the “find” function such that

```matlab
I1 = find(Dmask>0.01);
Cmask(I1) = 1;
```

The new mask is shown in Fig. 4.

![Filtered Map](image)

**Figure 4: Circuit with rounded edges.**

We will add colored noise to the foreground and background, each with different variances. This is called a noise mixture. The coloring is done with a Gaussian filter but the variance of the noises are different such that

```matlab
noise = rand(My,Nx);
sigma = 0.1;
hgaussnoise = gaperture(sigma,sigma,x,y);
Hgaussnoise = fft2(fftshift(hgaussnoise)); % shift to put origin at the corners
noiseF = real(ifft2(fft2(noise).*Hgaussnoise));
noiseF = noiseF-min(min(noiseF));
noiseF = noiseF/max(max(noiseF));
noiseF = noiseF;
noise0 = noiseF.*0.005;
noise1 = noiseF.*0.02;
I0 = find(Cmask<=0.1);
```
I1 = find(Cmask > 0.1);
Dmaskb4 = Dmask;
Dmask(I0) = Dmask(I0) + noise0(I0);
Dmask(I1) = Dmask(I1) + noise1(I1);

Figure 5: Surface with uniform noise on foreground and background surfaces.

Notice we saved the Dmaskb4 for later use. Noise0 is the background noise and Noise1 is the foreground noise. Both noises used in Fig. 5 are colored with a low pass Gaussian but the foreground has a higher variance leading to more roughness on the surface.

The next task adds realism to the model by using different colored noises for each surface that represents defects commonly seen in circuit boards. The defect on the “lines” are randomly placed cuts or indents in the circuit lines whereas the defects on the substrate is metallic splatter. The former leads to “opens” and the latter to “shorts” in the circuits operation. This model is appropriate for what is known as the additive process where the circuit lines are grown onto the substrate.

So, the next noise model not only incorporates homogeneous or “stationary” color noise but then through a thresholding process creates non-homogeneous “blobs” randomly spaced across the foreground and background regions.

% create red noise and thresh
% BACKGROUND NOISE BLOBS
sigma = 10;
x = 1:Nx;
x = x - Nx/2;
y = 1:My;
y = y - My/2;
hgauss2 = gaperture(sigma, sigma, x, y);
figure(6)
imagesc(hgauss2);
colormap gray;
axis image;
title('Background Colored Noise Filter')
print -djpeg fig6
Figure 6: Gaussian filter used to color background noise.

The resulting noise from the filter in Fig. after thresholding is

```matlab
Hgauss2=fft2(fftshift(hgauss2)); % shift to put origin at the corners
% background noise blobs
noise2=real(ifft2(fft2(noise).*Hgauss2));
noise2=noise2-min(min(noise2));
noise2=noise2/max(max(noise2));
eta=0.8;
I2=find(noise2<eta);
I2B=find((noise2>=eta)&(Cmask<=0.1)|(Dmask>=0.1));
noise2(I2)=eta;
noise2=noise2-eta;
noise2=noise2.*(1-Cmask);
```

Figure 7: Background defect noise.

The same is done for the foreground such that

```matlab
% FOREGROUND noise blobs
sigma=15;
x=1:Nx;
x=x-Nx/2;
y=1:My;
y=y-My/2;
hgauss3=gaperture(sigma,sigma,x,y);
```
Figure 8: Foreground defect color noise filter.

\[
\text{Hgauss3} = \text{fft2}(\text{fftshift(hgauss3))}; \quad \% \text{shift to put origin at the corners}
\]
\[
\text{noise3} = \text{real}(\text{ifft2}((\text{fft2(noise)}).*\text{Hgauss3}));
\]
\[
\text{noise3} = \text{noise3} - \min(\min(\text{noise3}));
\]
\[
\text{noise3} = \text{noise3} / \max(\max(\text{noise3}));
\]
\[
\eta = 0.3;
\]
\[
I3 = \text{find}(\text{noise3} > \eta);
\]
\[
\text{noise3}(I3) = \eta;
\]
\[
\text{noise3} = \text{noise3} - \eta;
\]
\[
\text{noise3} = \text{noise3} / \abs{\min(\min(\text{noise3}))};
\]
\[
\text{noise3} = \text{noise3} \times \text{Cmask};
\]
\[
\text{noise3} = 1 + \text{noise3};
\]

Figure 9: Foreground color noise blobs.

The two noise patterns in Figs 9 and 11 are mixed onto the surface as to modify the roughness and thus correspond to materials or processes involved in the defects such that:

\[
\% \text{mix in the noise}
\]
\[
Ib4 = \text{find}(((\text{Dmaskb4} > 0.1) \& (\text{noise3} < 1)) \mid ((\text{Dmaskb4} > 0.1) \& (\text{noise2} > 0)));
\]
\[
\text{Dmask}(Ib4) = \text{Dmaskb4}(Ib4);
\]
Dmask=Dmask+noise2; % use additive noise to position spatter on substrate
Dmask=Dmask.*noise3; % use multiplication to cut away circuit lines

Figure 10: 3-D surface before color mapping.

The last step is to color the surfaces and store in a 3D format. The software needed and
description of formats can be found in the references. The GL3Dviewer can export the data in
VRML which can be viewed in programs like MESHLAB and BLENDER. The coloring and
storage in mat5 format is as follows:

% create a mat5 3D surface
Albedo8=uint8(zeros(My,Nx));
% gold color 255, 215, 0
% copper color 184,115,51
imageC=uint8(zeros(My,Nx,3));
% red
Albedo8(I0)=184;
Albedo8(I1)=255;
Albedo8(I2B)=255;
imageC(:,:,1)=Albedo8;
% green
Albedo8(I0)=115;
Albedo8(I1)=215;
Albedo8(I2B)=215;
imageC(:,:,2)=Albedo8;
% blue
Albedo8(I0)=51;
Albedo8(I1)=0;
Albedo8(I2B)=0;
imageC(:,:,3)=Albedo8;
% make all pixels high quality value of 20
imageI=uint8(20*ones(My,Nx,3));
index=1;
xw=zeros(My,Nx);
yw=xw;
zw=yw;
zw=Dmask*dz;
for x=1:Nx
for y=1:My
    xw(index)=(x-1)*dx;
    yw(index)=(My-y)*dy;
    index=index+1;
end;
mat5prefix = 'Circuits3D';
result= mat5CIXYZwrite2011(mat5prefix,imageC,imageI,xw,yw,zw);

Using the GL3Dviewer and using the setting in the

![3D rendering of surface](image)

**Figure 11: 3D rendering of surface.**

Figure 11 shows the final result.

**REFERENCES**

1. MAT5 Format: [http://www.engr.uky.edu/~lgh/soft/softmat5format.htm](http://www.engr.uky.edu/~lgh/soft/softmat5format.htm)
2. GL3Dview.exe and Tutorials: [http://www.engr.uky.edu/~lgh/soft/softhowtoGL3DView.htm](http://www.engr.uky.edu/~lgh/soft/softhowtoGL3DView.htm)