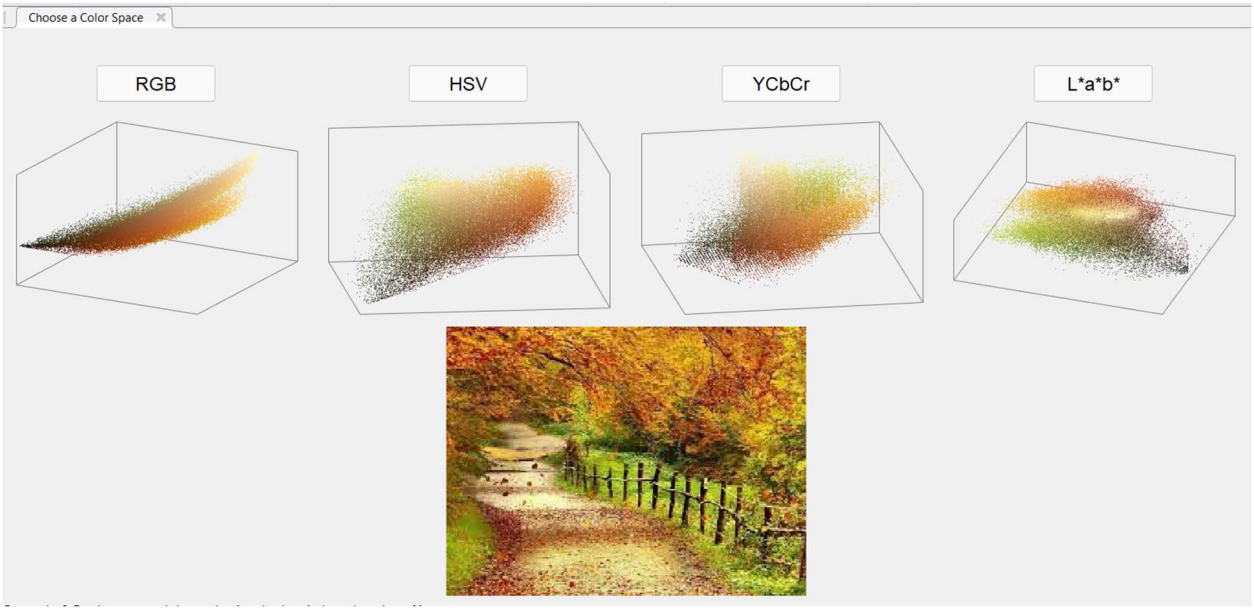
DIP LAB 9

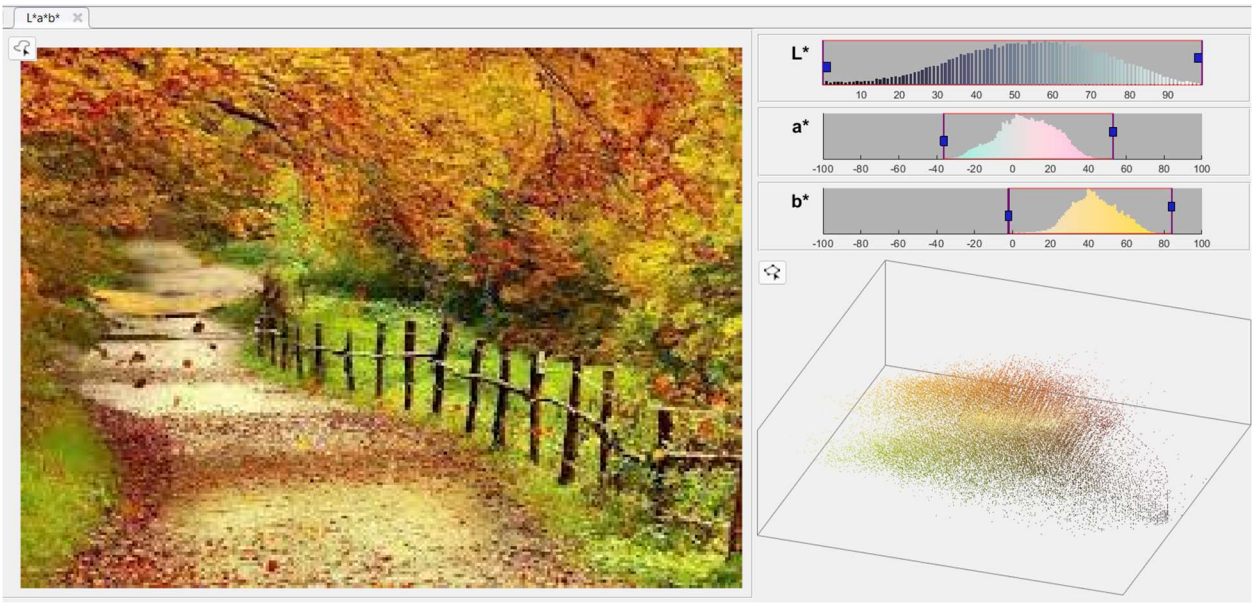
V.M. Bhuvanesh
19BCD7088

Image Segmentation using thresholding

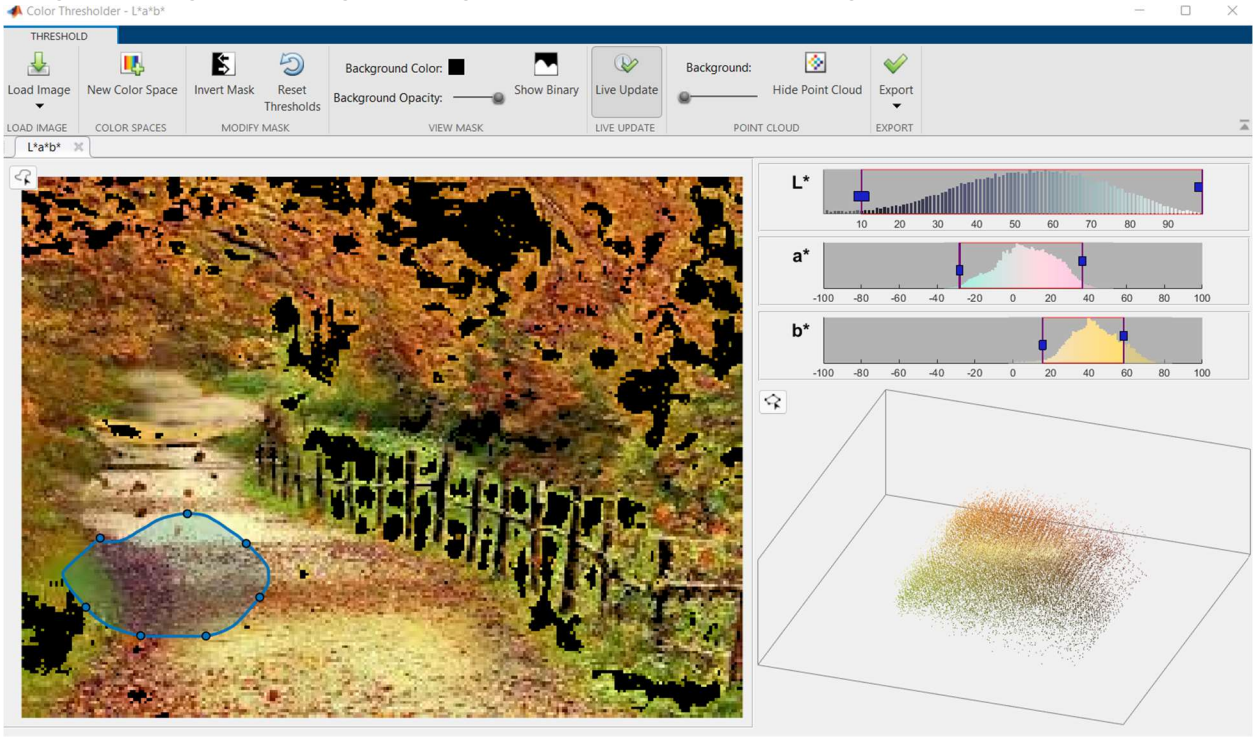
Original image:

Color spaces shown by image:

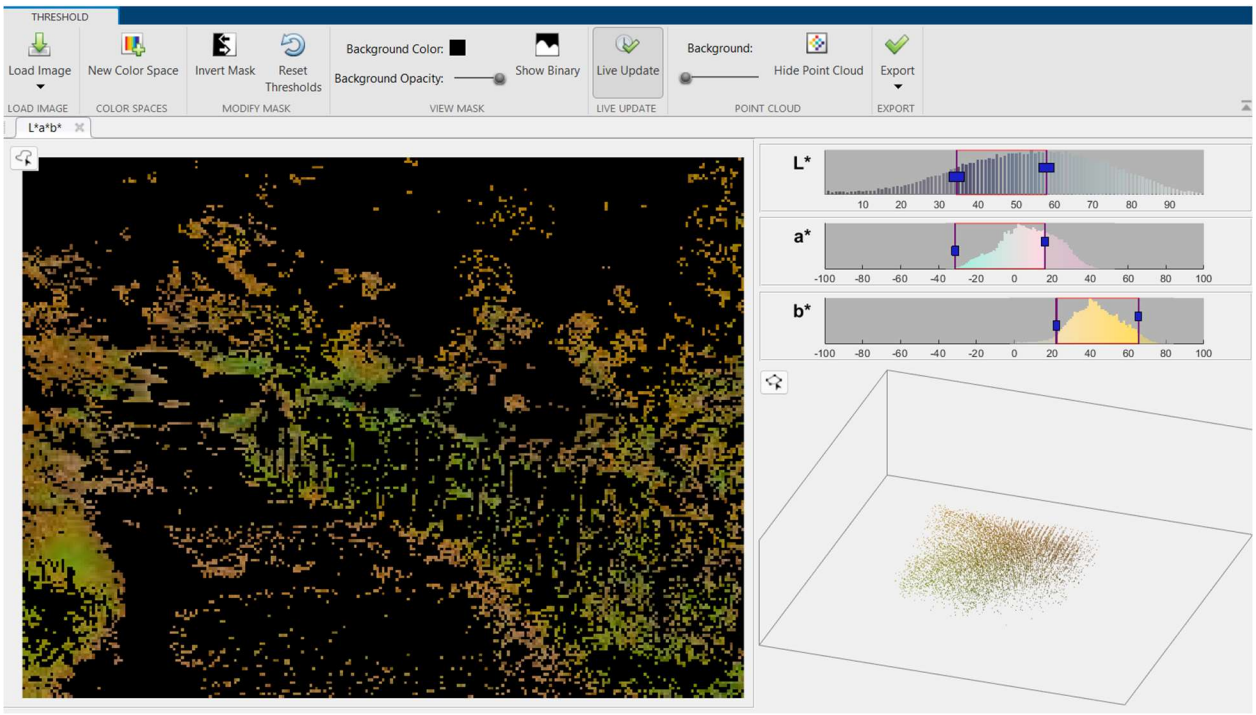




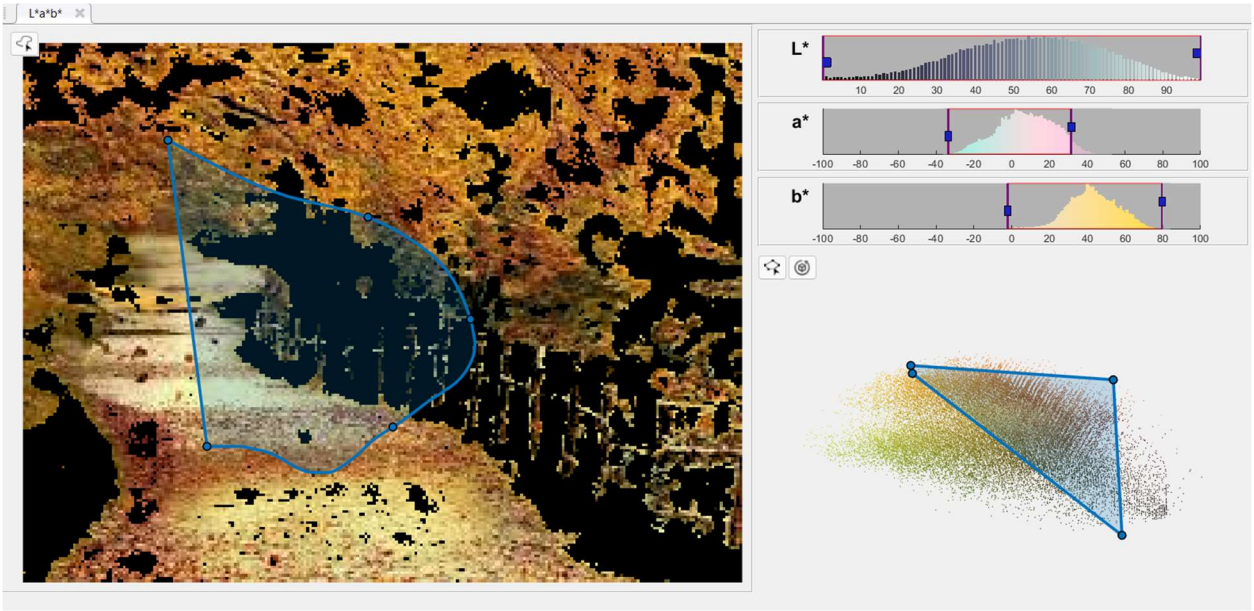
Segmenting the image using automatic thresholding:



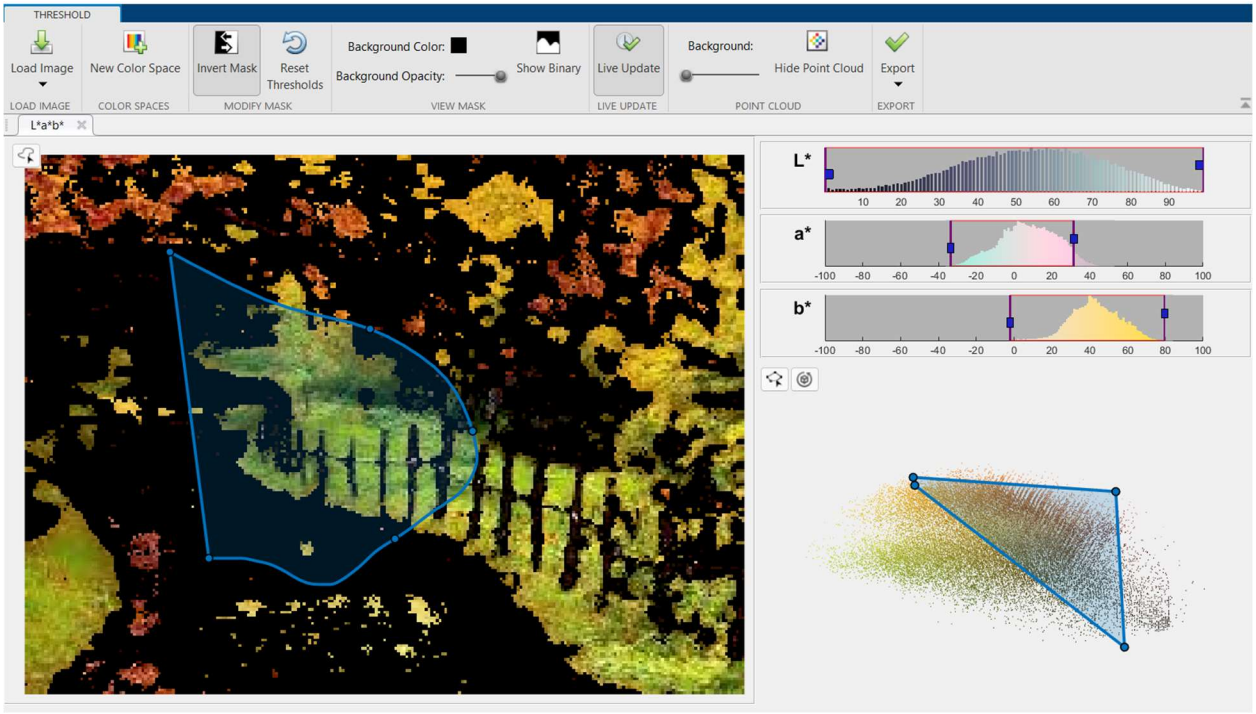
Refining automatic thresholding using color controls:



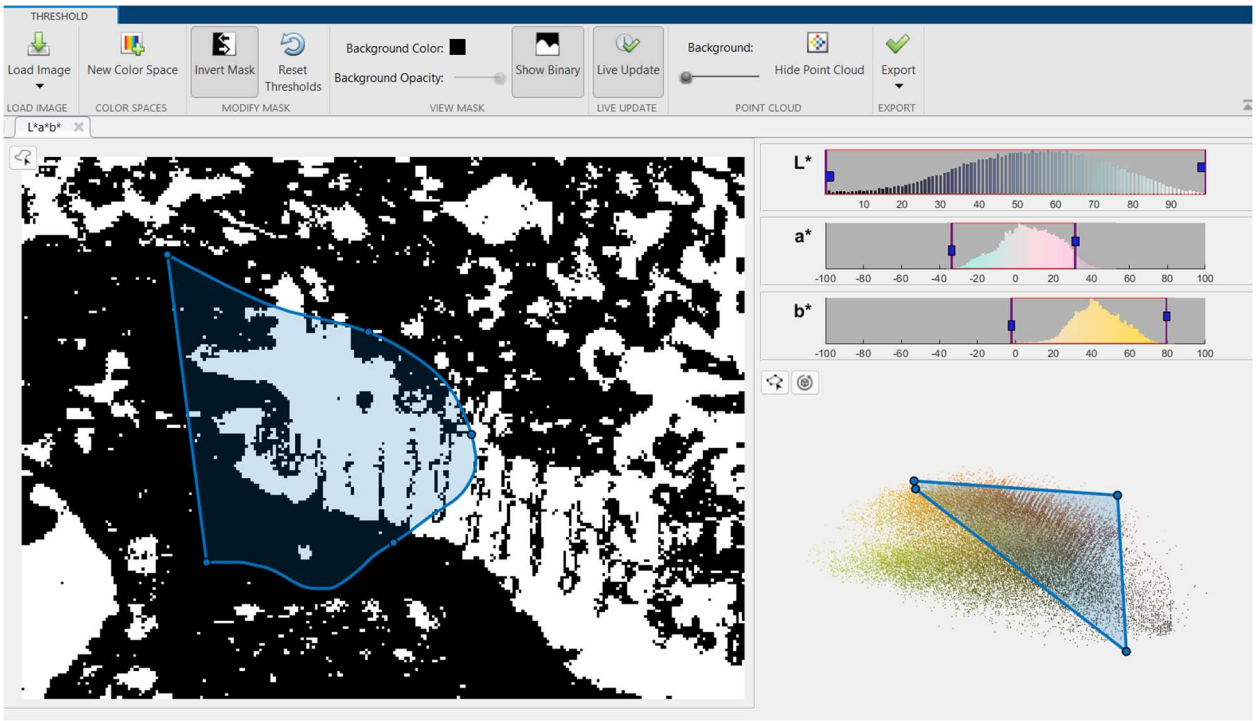
Thresholding image color using point cloud:



Invert mask:


THRESHOLD
Load Image
New Color Space
Invert Mask
Reset Thresholds
Background Color:
Background Opacity:
Show Binary
Live Update
Background:
Hide Point Cloud
Export
LOAD IMAGE
COLOR SPACES
MODIFY MASK
VIEW MASK
LIVE UPDATE
POINT CLOUD
EXPORT
L*a*b*
L*
a*
b*

Binary mask:


THRESHOLD
Load Image
New Color Space
Invert Mask
Reset Thresholds
Background Color:
Background Opacity:
Show Binary
Live Update
Background:
Hide Point Cloud
Export
LOAD IMAGE
COLOR SPACES
MODIFY MASK
VIEW MASK
LIVE UPDATE
POINT CLOUD
EXPORT
L*a*b*
L*
a*
b*

Code required to recreate the segmentation:

lab2.m | lab3.m | lab4.m | lab5.m | lab5b.m | lab5c.m | lab5c2.m | lab5d.m | lab6.m | lab7.m | lab8.m | untitled

```
function [BW,maskedRGBImage] = createMask(RGB)
%createMask  Threshold RGB image using auto-generated code from colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace and
%  range for each channel of the colorspace were set within the app. The
%  segmentation mask is returned in BW, and a composite of the mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 12-May-2022
%------------------------------------------------------

RGB=imread("nature.jpg");
% Convert RGB image to chosen color space
I = rgb2lab(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.196;
channel1Max = 98.960;

% Define thresholds for channel 2 based on histogram settings
channel2Min = -33.493;
channel2Max = 31.552;

% Define thresholds for channel 3 based on histogram settings
channel3Min = -2.092;
channel3Max = 79.544;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
```

Command Window

```
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);

% Create mask based on selected regions of interest on point cloud projection
I = double(I);
[m,n,~] = size(I);
polyBW = false([m,n]);
I = reshape(I,[m*n 3]);
temp = I(:,1);
I(:,1) = I(:,2);
I(:,2) = I(:,3);
I(:,3) = temp;
clear temp

% Project 3D data into 2D projected view from current camera view point within app
J = rotateColorSpace(I);

% Apply polygons drawn on point cloud in app
polyBW = applyPolygons(J,polyBW);

% Combine both masks
BW = sliderBW & polyBW;

% Invert mask
BW = ~BW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;
```

```
end

function J = rotateColorSpace(I)

% Translate the data to the mean of the current image within app
shiftVec = [7.467597 43.019465 54.130602];
I = I - shiftVec;
I = [I ones(size(I,1),1)]';

% Apply transformation matrix
tMat = [0.003618 -0.010458 -0.000000 0.307123;
    0.008955 0.003159 0.004643 -0.801875;
    0.005200 0.001835 -0.007997 8.772824;
    0.000000 0.000000 0.000000 1.000000];

J = (tMat*I)';
end

function polyBW = applyPolygons(J,polyBW)

% Define each manually generated ROI
hPoints(1).data = [0.067122 -0.530150;
    0.592253 -0.598556;
    0.614817 -1.343426;
    0.071225 -0.568153];

% Iteratively apply each ROI
for ii = 1:length(hPoints)
    if size(hPoints(ii).data,1) > 2
```

```
J = (tMat*I)';
end

function polyBW = applyPolygons(J,polyBW)

% Define each manually generated ROI
hPoints(1).data = [0.067122 -0.530150;
    0.592253 -0.598556;
    0.614817 -1.343426;
    0.071225 -0.568153];

% Iteratively apply each ROI
for ii = 1:length(hPoints)
    if size(hPoints(ii).data,1) > 2
        in = inpolygon(J(:,1),J(:,2),hPoints(ii).data(:,1),hPoints(ii).data(:,2));
        in = reshape(in,size(polyBW));
        polyBW = polyBW | in;
    end
end

end
```

Editor - D:\6th sem\Digital Image Processing\lab\bhuvcreateMask.m

lab2.m | lab3.m | lab4.m | lab5.m | lab5b.m | lab5c.m | lab5c2.m | lab5d.m | lab6.m | lab7.m | lab8.m | untitled8.m | vm.m | vysh.m | bhuv.m | bhuvcreateMask.m

```
function [BW,maskedRGBImage] = createMask(RGB)
%createMask  Threshold RGB image using auto-generated code from colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace and
%  range for each channel of the colorspace were set within the app. The
%  segmentation mask is returned in BW, and a composite of the mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 12-May-2022
%------------------------------------------------------

RGB=imread("nature.jpg");
% Convert RGB image to chosen color space
I = rgb2lab(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.196;
channel1Max = 98.960;

```

Command Window

```
     0     0     0     0     1     0     0
     1     0     0     0     0     0     0
     1     0     0     1     0     0     0
     1     0     0     1     1     0     0
     1     0     0     1     0     0     0
     0     0     0     1     0     0     0
     1     0     0     0     0     0     0
     1     0     0     0     0     0     0
     0     0     0     1     0     0     0
fx   1     0     0     1     0     0     1
```

Zoom: 110% | UTF-8 | CRLF | createMask | Ln 12