
Subject: Help converting MATLAB to IDL

Posted by [mikejohnryan08](#) on Wed, 16 Apr 2014 00:14:55 GMT

[View Forum Message](#) <> [Reply to Message](#)

I am trying to run a previously written MATLAB script to create a 3d anaglyph from two images in FreeMat, but it is very slow. I've recently received a temporary license for IDL 8.3, which I would love to replicate this code in, as I believe both languages are capable of doing this.

But since I am still very unfamiliar with IDL I am having difficulty doing so. Any advice/help for converting this would be fantastic! Here is the MATLAB code, followed by my attempt at conversion to IDL:

```
function anaglyph=MCAA(lImg,rImg)
%=====
=====
%Copyright(c) 2013 Li Songnan
%All Rights Reserved.
%
%This is an implementation of the anaglyph image generation algorithm described in
%the following paper:
%
%[1] S. Li, L. Ma, K.N. Ngan, "Anaglyph Image Generation by Matching Color
%Appearance Attributes", Signal Processing - Image Communication, accepted, 2013
%
%Input : (1) lImg: left image
%        (2) rImg: right image
%Output: (1) anaglyph: anaglyph image
%=====
=====

% Register images together

% Change range of pixel value from [0,255] to [0,1]
lImg=double(lImg)./255;
rImg=double(rImg)./255;

% Forward gamma transform
lImg2 = gamma_forward(lImg);
rImg2 = gamma_forward(rImg);

% Reshape images into a 3xN matrix (N is the total pixel number)
[h,w,tmp]=size(rImg2);
lImgNx3=reshape(lImg2,[h*w,3]);
rImgNx3=reshape(rImg2,[h*w,3]);
lImg3xN=double(lImgNx3');
rImg3xN=double(rImgNx3');

% RGB-CIELAB conversion matrix
```

```

C=[0.4243 0.3105 0.1657;...      % LCD
    0.2492 0.6419 0.1089;...
    0.0265 0.1225 0.8614];
Ar=[0.0153 0.1092 0.1171;...      % Cyan lens
    0.0176 0.3088 0.0777;...
    0.0201 0.1016 0.6546];
Al=[0.1840 0.0179 0.0048;...      % Red lens
    0.0876 0.0118 0.0018;...
    0.0005 0.0012 0.0159];

% Processing right image
GB2xN=process_right_img(rImg3xN,C,Ar);

% Procssing left image
R1xN=process_left_img(lImg3xN,GB2xN,C,Al);

% Combine RGB
RGB3xN=[R1xN;GB2xN];

% Reshape a 3xM matrix to an image
RGBNx3=RGB3xN';
anaglyph=reshape(RGBNx3,[h,w,3]);

% Backward gamma transform
anaglyph=gamma_backward(anaglyph);

% Change range of pixel value back to [0,255]
anaglyph=uint8(anaglyph.*255);
end

function GB=process_right_img(img,C,Ar)

% Convert RGB to LAB
[L, a, b] = RGB2LAB(img, C);

% Calculate hue and saturation
[H,S]=get_hue_saturation(a,b);

% Calculate L
L2=get_L(L,H,S);

% Calculate a and b
[a2,b2]=get_AB(H,S);

% Convert LAB to RGB
RGB = LAB2RGB(L2,a2,b2,Ar);

% Use G and B only

```

```

    GB=RGB(2:3,:);
    GB(:)=clip_0_1(GB(:));
end

function R=process_left_img(img,GB,C,Al)
    % Convert RGB to LAB
    L = RGB2LAB(img, C);

    % Implementation of eqs. (18) and (19)
    Yn=Al(2,:)*[1;1;1];
    Y=(finv((16+L)/116))*Yn;
    R=max((Y-Al(2,2)*GB(1,:)-Al(2,3)*GB(2,:)),0)/Al(2,1);
end

function [H, S]=get_hue_saturation(a, b)
    % Implementation of eqs. (6) and (7)
    H=(180/pi)*atan(b./(a+1e-10))+180*(a<0);
    S=sqrt(a.^2+b.^2);
end

function L2=get_L(L,H,S)
    L2=L;

    % Reduce lightness of red-like colors
    Hred=41.6;
    T=15;
    hueAbsDiff=abs(H-Hred);
    pos=find(abs(H-Hred)<=T);
    L2(pos)=tune_lightness(L(pos),hueAbsDiff(pos),T,S(pos));
end

function L2=tune_lightness(L,hueAbsDiff,T,S)
    Sl = 40;
    Sh = 50;
    Pmax = 0.4;

    % Implementation of eq. (13)
    Psr = Pmax*ones(size(L));
    Psr(find(S<Sl))=0;
    pos=find(S>=Sl&S<=Sh);
    Psr(pos)=Pmax*((S(pos)-Sl)/(Sh-Sl));

    % Implementation of eq. (12)
    L2=(1-Psr.*(T-hueAbsDiff)/T).*L;
end

function [a2,b2]=get_AB(H,S)
    Hred=41.6;

```

```

Hcyan=221.6;
T=15;

% Adjust the saturation
hueAbsDiff=min(abs(H-Hred),abs(H-Hcyan));
pos=find(hueAbsDiff<=T);
S(pos)=S(pos).*(hueAbsDiff(pos)/T);

% Project to the green region
pos1=find(H>Hred & H<Hcyan);
[a2(pos1),b2(pos1)]=project_circle(S(pos1));

% Project to the blue region
pos2=find(H<=Hred | H>=Hcyan);
[a2(pos2),b2(pos2)]=project_line(S(pos2));
end

function [a,b]=project_circle(S)
    Ca=125;
    Cb=172;
    r=S;
    a=(4*Ca*r.^2-4*Cb.*r.*sqrt(4*Ca^2+4*Cb^2-r.^2))/(8*Ca^2+8*Cb ^2);
    b=sqrt(r.^2-a.^2);
end

function [a,b]=project_line(S)
    k=-0.7273;
    r=S;
    a=sqrt(r.^2/(1+k^2));
    b=k*a;
end

function [L, a, b] = RGB2LAB(img, C)
    % Convert RGB to XYZ
    XYZ=C*img;
    X=XYZ(1,:);
    Y=XYZ(2,:);
    Z=XYZ(3,:);

    % Calculate XYZ of the white point
    white=[1;1;1];
    XnYnZn=C*white;
    Xn=XnYnZn(1);
    Yn=XnYnZn(2);
    Zn=XnYnZn(3);

    % Convert XYZ to LAB
    L=116*f(Y./Yn)-16;

```

```

a=500*(f(X./Xn)-f(Y./Yn));
b=200*(f(Y./Yn)-f(Z./Zn));
end

```

```

function RGB = LAB2RGB(L,a,b,Ar)
% Calculate XYZ of the white point
XnYnZn=Ar*[1;1;1];
Xn=XnYnZn(1);
Yn=XnYnZn(2);
Zn=XnYnZn(3);

% Convert LAB to XYZ
X=Xn*finv((L+16)/116+a/500);
Y=Yn*finv((L+16)/116);
Z=Zn*finv((L+16)/116-b/200);
XYZ=[X;Y;Z];

% Convert XYZ to RGB
RGB=inv(Ar)*XYZ;
end

```

```

function img2=gamma_forward(img)
img2=zeros(size(img));
pos=find(img>0.04045);
img2(pos)=((img(pos)+0.055)./1.055).^2.4;
pos=find(img<=0.04045);
img2(pos)=img(pos)./12.92;
end

```

```

function img2=gamma_backward(img)
img2=zeros(size(img));
pos=find(img<=0.0031308);
img2(pos)=12.92.*img(pos);
pos=find(img>0.0031308);
img2(pos)=1.055.*(img(pos).^0.41666)-0.055;
end

```

```

function w=f(w)
pos1=find(w>0.008856);
pos2=find(w<=0.008856);
w(pos1)=w(pos1).^(1/3);
w(pos2)=7.787*w(pos2)+(16/116);
end

```

```

function w=finv(w)
pos1=find(w>6/29);

```

```

pos2=find(w<=6/29);
w(pos1)=w(pos1).^(3);
w(pos2)=3*((6/29)^2)*(w(pos2)-(4/29));
end

```

```

function matrix=clip_0_1(matrix)
    matrix=min(max(0,matrix),1);
end

```

MY IDL ATTEMPT:

```

function anaglyph, l_img, r_img

; Change pixel value range from [0,255] to [0,1]
l_img = double(l_img)/255
r_img = double(r_img)/255

; Gamma transform
l_img2 = gamma_forward(l_img)
r_img2 = gamma_forward(r_img)

; Reshape image into 3xN matrix (N is total pixel number)
[h,w,tmp] = size(r_img2)
l_imgNx3 = reform(l_img2, [h*w,3])
r_imgNx3 = reform(r_img2, [h*w,3])
l_img3xN = double(l_imgNx3)
r_img3xN = double(r_imgNx3)

; RGB-CIELAB conversion matrix
C=[[0.4243 0.3105 0.1657], ; LCD
[0.2492 0.6419 0.1089],
[0.0265 0.1225 0.8614]]
Ar=[[0.0153 0.1092 0.1171], ; Cyan lens
[0.0176 0.3088 0.0777],
[0.0201 0.1016 0.6546]]
Al=[[0.1840 0.0179 0.0048], ; Red lens
[0.0876 0.0118 0.0018],
[0.0005 0.0012 0.0159]]

; Processing right image
GB2xN=process_right_img(rImg3xN,C,Ar)

; Procssing left image
R1xN=process_left_img(lImg3xN,GB2xN,C,Al);

```

```

; Combine RGB
RGB3xN=[R1xN,GB2xN]

; Reshape a 3xM matrix to an image
RGBNx3=transpose(RGB3xN)
anaglyph=reform(RGBNx3,[h,w,3])

; Backward gamma transform
anaglyph=gamma_backward(anaglyph)

; Change range of pixel value back to [0,255]
anaglyph=bytsc1(anaglyph)

return, anaglyph

end

```

```

function process_right_img(img,C,Ar)

; Convert RGB to LAB
[L, a, b] = RGB2LAB(img, C)

; Calculate hue and saturation
[H,S]=get_hue_saturation(a,b)

; Calculate L
L2=get_L(L,H,S)

; Calculate a and b
[a2,b2]=get_AB(H,S)

; Convert LAB to RGB
RGB = LAB2RGB(L2,a2,b2,Ar)

; Use G and B only
GB=RGB(2:3,:)
GB(:)=clip_0_1(GB(:))

return, GB

end

```

```

function process_left_img(img,GB,C,Al)
; Convert RGB to LAB
L = RGB2LAB(img, C)

; Implementation of eqs. (18) and (19)
Yn=Al(2,:)*[[1],[1],[1]]

```

```

Y=(finv((16+L)/116))*Yn;
R=max((Y-AI(2,2)*GB(1,:)-AI(2,3)*GB(2,:),0)/AI(2,1);

return, R, L

end

function get_hue_saturation(a, b)
; Implementation of eqs. (6) and (7)
H=(180/pi)*atan(b./(a+1e-10))+180*(a<0)
S=sqrt(a.^2+b.^2)

return, H, S

end

function get_L(L,H,S)
L2=L

; Reduce lightness of red-like colors
Hred=41.6
T=15
hueAbsDiff=abs(H-Hred)
pos=find(abs(H-Hred)<=T)
L2(pos)=tune_lightness(L(pos),hueAbsDiff(pos),T,S(pos))

return, L2

end

function tune_lightness(L,hueAbsDiff,T,S)
Sl = 40
Sh = 50
Pmax = 0.4

; Implementation of eq. (13)
Psr = Pmax*ones(size(L))
Psr(find(S<Sl))=0
pos=find(S>=Sl&S<=Sh)
Psr(pos)=Pmax*((S(pos)-Sl)/(Sh-Sl))

; Implementation of eq. (12)
L2=(1-Psr.*(T-hueAbsDiff)/T).*L

return, L2

end

```



```

function =get_AB(H,S)
    Hred=41.6
    Hcyan=221.6
    T=15

    ; Adjust the saturation
    hueAbsDiff=min(abs(H-Hred),abs(H-Hcyan))
    pos=find(hueAbsDiff<=T)
    S(pos)=S(pos).*(hueAbsDiff(pos)/T)

    ; Project to the green region
    pos1=find(H>Hred & H<Hcyan)
    [a2(pos1),b2(pos1)]=project_circle(S(pos1))

    ; Project to the blue region
    pos2=find(H<=Hred | H>=Hcyan)
    [a2(pos2),b2(pos2)]=project_line(S(pos2))

    return, a2, b2

end

function project_circle(S)
    Ca=125
    Cb=172
    r=S
    a=(4*Ca*r.^2-4*Cb.*r.*sqrt(4*Ca^2+4*Cb^2-r.^2))/(8*Ca^2+8*Cb ^2)
    b=sqrt(r.^2-a.^2)

    return, a, b

end

function project_line(S)
    k=-0.7273
    r=S
    a=sqrt(r.^2/(1+k^2))
    b=k*a

    return, a, b

end

function RGB2LAB(img, C)
    ; Convert RGB to XYZ
    XYZ=C*img
    X=XYZ(1,:)
    Y=XYZ(2,:)

```

```

Z=XYZ(3,:)

; Calculate XYZ of the white point
white=[[1],[1],[1]]
XnYnZn=C*white
Xn=XnYnZn(1)
Yn=XnYnZn(2)
Zn=XnYnZn(3)

; Convert XYZ to LAB
L=116*(Y./Yn)-16
a=500*(f(X./Xn)-f(Y./Yn))
b=200*(f(Y./Yn)-f(Z./Zn))

return, L, a, b

end

```

```

function LAB2RGB(L,a,b,Ar)
; Calculate XYZ of the white point
XnYnZn=Ar*[[1],[1],[1]]
Xn=XnYnZn(1)
Yn=XnYnZn(2)
Zn=XnYnZn(3)

; Convert LAB to XYZ
X=Xn*finv((L+16)/116+a/500)
Y=Yn*finv((L+16)/116)
Z=Zn*finv((L+16)/116-b/200)
XYZ=[[X],[Y],[Z]]

; Convert XYZ to RGB
RGB=inv(Ar)*XYZ

return, RGB

end

```

```

function gamma_forward(img)
img2=zeros(size(img))
pos=find(img>0.04045)
img2(pos)=((img(pos)+0.055)/1.055).^2.4
pos=find(img<=0.04045)
img2(pos)=img(pos)/12.92

return, img2

end

```

```
function gamma_backward(img)
    img2=zeros(size(img))
    pos=find(img<=0.0031308)
    img2(pos)=12.92.*img(pos)
    pos=find(img>0.0031308)
    img2(pos)=1.055.*(img(pos).^0.41666)-0.055
```

```
    return, img2
```

```
end
```

```
function f(w)
    pos1=find(w>0.008856)
    pos2=find(w<=0.008856)
    w(pos1)=w(pos1).^(1/3)
    w(pos2)=7.787*w(pos2)+(16/116)
```

```
    return, w
```

```
end
```

```
function finv(w)
    pos1=find(w>6/29)
    pos2=find(w<=6/29)
    w(pos1)=w(pos1).^(3)
    w(pos2)=3*((6/29)^2)*(w(pos2)-(4/29))
```

```
    return, w
```

```
end
```

```
function clip_0_1(matrix)
    matrix=min(max(0,matrix),1)
```

```
    return, matrix
```

```
end
```
