Color Filters to Enhance/Reduce Color Perception Contrast - Pickleball Problem - Priyanka Dilip, Xin-Yi Pan, Rita Meraz, Maryann Benny Fernandes

Team Members: Priyanka Dilip, Xin-Yi Pan, Rita Meraz, Maryann Benny Fernandes

Introduction

1. **The Problem**

The craze to play pickleball is increasing linearly but where are the courts? As illustrated in figure 1, we see frustrated tennis players watch an optimistic yet relaxed pickleball player add blue pickleball lines to their tennis court.

Figure 1: A pickleball player adapting a tennis court to a pickleball court with blue tape while angry tennis players watch the action

A report published by the San Francisco Standard news organization in the Art and Culture section, "Tensions Run High as Tennis and Pickleball Players Fight Over Future of Stern Grove Courts" clearly depicts the frustration behind tennis courts getting adapted to pickleball courts. For example, in figure 2, we see a tennis court with four pickleball courts, which for any tennis player would be frustrating to play on because now from two colors, the white lines and the green surface, they now need to concentrate on four different colors, including the pickleball brown court and yellow lines.

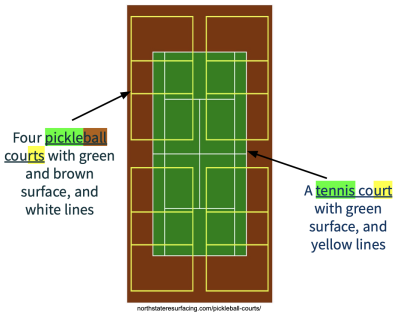


Figure 2: A tennis court with blue lines on a green surface with four pickleball courts with yellow lines on a green and brown surface

2. **The Solution**

In this report, we present tangible solutions to tackle the problem of a pickleball court adapted to a tennis court. We use the idea of the color-matching technique in the engineering of image systems. This technique helps us understand which color of light in the visible light range of the electromagnetic spectrum appears the same to a human eye, or a camera, even when their physical properties are completely different. It is also possible for us to use the principles of color matching to create special visual effects as illustrated in figure 3. For example, we can design glasses with color filters that make certain objects appear more similar, and certain objects that appear similar appear different.

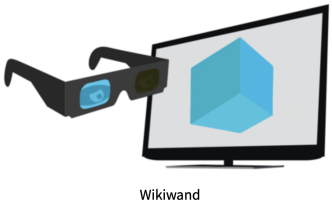


Figure 3: Color matching to create special visual effects

Background

To study a case of a pickleball court adapted to a tennis court, our team along with the teaching staff of EE 221, as illustrated in figure 3, went on a field trip to Alpine Hills Tennis Club at Portola Valley, California as seen in figure 4 and used court 11 for data capturing.

Figure 3: From left to right, David Cardinal, Brian Wandell, Joyce Farrell, Xin-Yi Pan, Priyanka Dilip, Maryann Benny Fernandes, and Rita Meraz

Figure 4: Court 11, Alpine Hills Tennis Club at Portola Valley, California

Using a spectroradiometer, we captured five different types of scenes when in sunlight and in the shade as mentioned in table 1. Figure 5, depicts scene 1 of the table, the spectroradiometer is capturing light measurements in wavelength and amplitude from a white tennis line on the Pickleball court when in sunlight , and in figure 6, when in shade, considering three cases the surface, a white calibrator, and a tennis ball.

Figure 1: Spectroradiometer Data Capturing Scenes on Court 11

	Spectroradiometer Targets	In sunlight	In shade
1	Tennis court lines on Pickleball court	yes	yes
2	Pickleball court with blue background	yes	yes
3	Pickleball court with green background	yes	yes
4	Blue Pickleball lines on green background	yes	yes
5	Blue Pickleball lines on blue background	yes	-

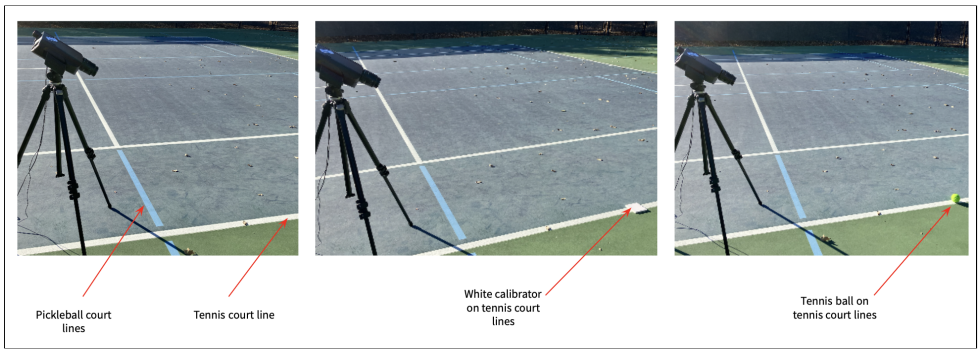


Figure 5: Tennis ball Lines on Pickleball court, white calibrator, and tennis ball in sunlight

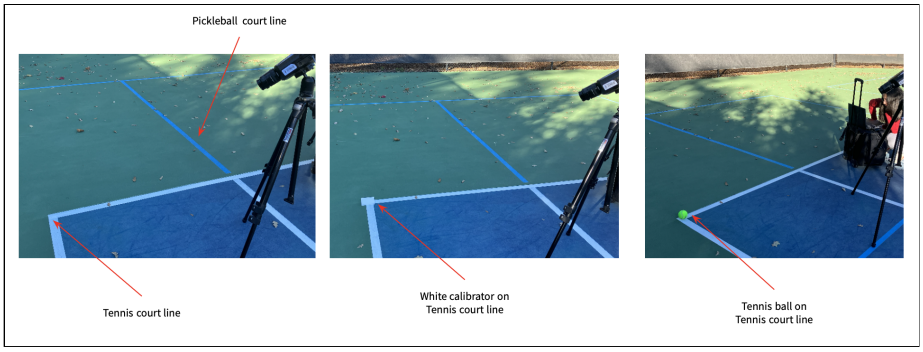


Figure 6: Tennis ball Lines on Pickleball court, white calibrator, and tennis ball in shade

Figure 7, depicts scene 2 of the table, the spectroradiometer is capturing light measurements in wavelength and amplitude from the blue background of the Pickleball court when in sunlight and in figure 8, when in shade.

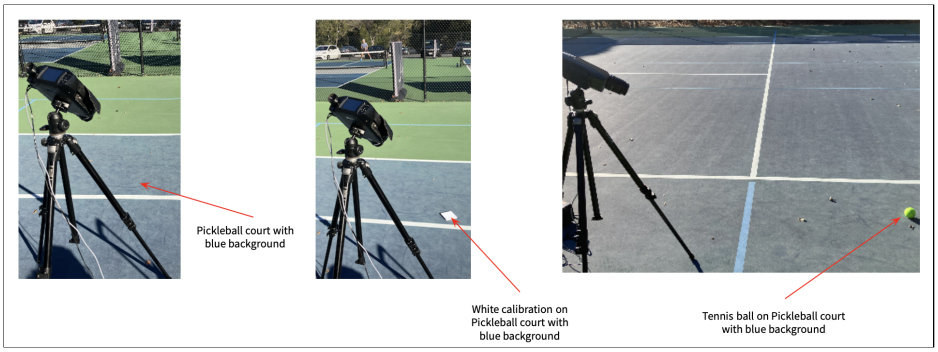


Figure 7: Pickleball court with a blue background, white calibrator, and tennis ball in sunlight

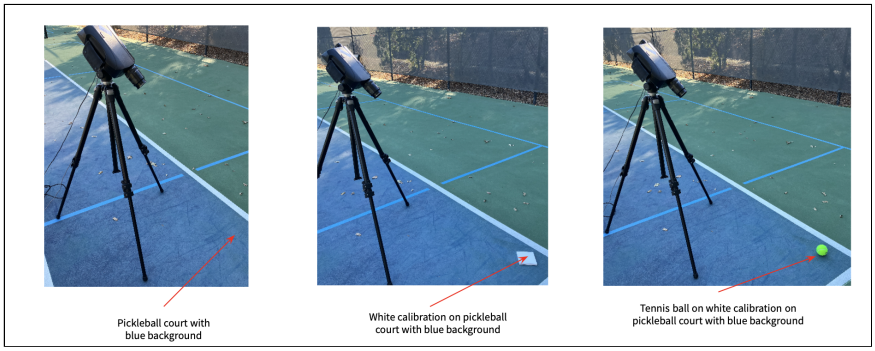


Figure 8: Pickleball court with a blue background, white calibrator, and tennis ball in shade

Figure 9, depicts scene 3 of the table, the spectroradiometer is capturing light measurements in wavelength and amplitude from the green background of the Pickleball court when in sunlight and in figure 10, when in shade.

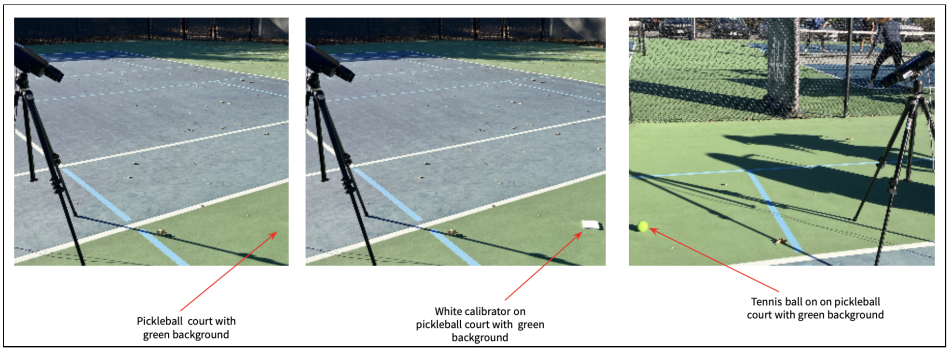


Figure 9: Pickleball court with a green background, white calibrator, and tennis ball in sunlight

Figure 10: Pickleball court with a green background, white calibrator, and tennis ball in shade

Figure 11, depicts scene 4 of the table, the spectroradiometer is capturing light measurements in wavelength and amplitude from the Pickleball court blue lines when it is on the green background in sunlight and in figure 12, when in shade.

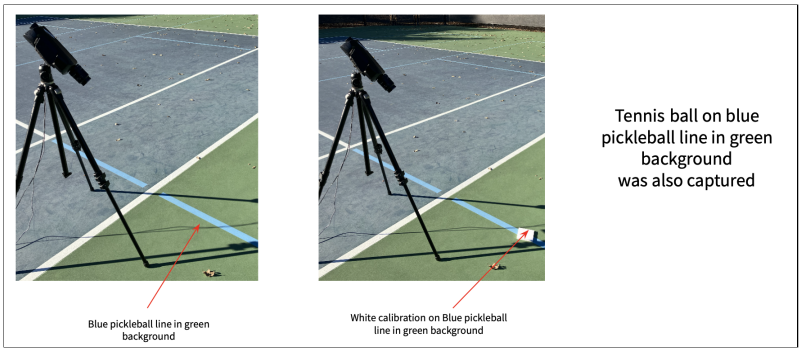


Figure 11: Blue pickleball lines on green background, and white calibrated in sunlight

Figure 12: Figure 11: Blue pickleball lines on green background, and white calibrated in shade

Figure 13, depicts scene 4 of the table, the spectroradiometer is capturing light measurements in wavelength and amplitude from the Pickleball court blue lines when it is on a blue background in sunlight.

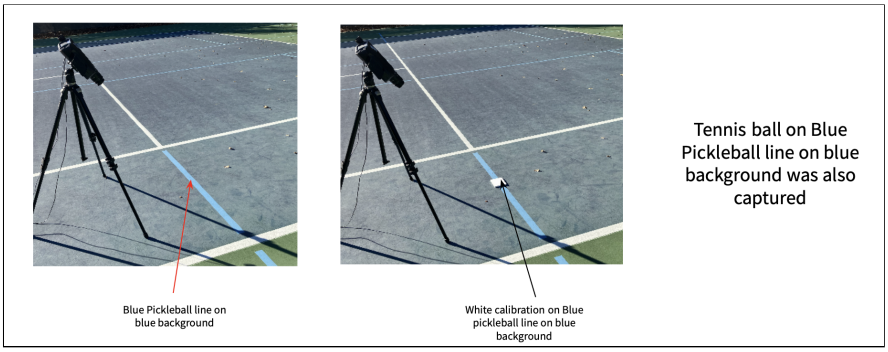


Figure 13: Blue pickleball lines on a blue background, and white calibrater in sunlight

Now, after having captured all our desired scenes on the Pickleball court, as seen in figure 14, we tried to load the data for both cases when in sunlight and in shade, added a filter, and iterated it to establish our desired results as mentioned in our methods and results section of this report.

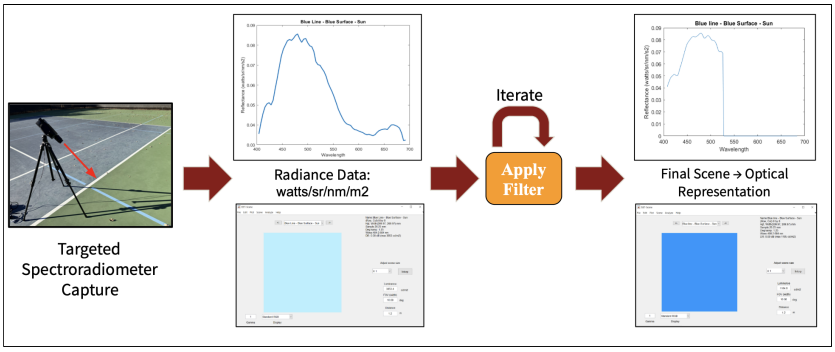


Figure 14: Block diagram of the solution - Color Filters to Enhance/Reduce Color Perception Contrast

Methods

1. **Leveraging isetCam**
    a. **xyz2lms :** Transform an XYZ image to Stockman LMS cone format

    imgLMS = xyz2lms(imgXYZ, [cbType], [method], [varargin])

    lms = **xyz2lms**(imgXYZ, 0, 'Brettel', whiteXYZ);

    Cbtype = 0 Default. Means a trichromatic observer.

    Method = brettel: Default. Use Bettel's algorithm. Provide numeric value of a white XYZ value.

    A calculation for color blind dichromats can also be performed by changing the cbtype

    The imgXYZ is the XYZ image to transform.

    a. Then, we use **colorTransformMatrix** which returns a color matrix that is suitable for use with imageLinearTransform, to convert an RGB (NxMx3) color image from one color space to another.

    imageLinearTransform: Apply a linear transformation to the channels of an RGB or XW image

    sRGB is a standard RGB color space to use on monitors, printers, etc.

    Xyz2srgb: Convert CIE XYZ to sRGB color space. The CIE XYZ values are in an RGB Format image. They are converted to sRGB values. scale the XYZ image so that it is within the [0, 1] range as required by the sRGB standard.

    The sRGB color space is a display-oriented representation that matches a Sony Trinitron. The monitor white point is assumed to be D65. The white point chromaticity are (.3127, .3290), and for an sRGB display (1, 1, 1) is assumed to map to XYZ = (0.9504 0.9999 1.0891).

    This xyz -> sRGB matrix is supposed to work for XYZ values scaled so that the maximum Y value is around 1.

    oiCreate: The optical image represents the spectral irradiance at the sensor. The irradiance is computed from the scene radiance, using the information in the optics structure that is attached to this oi.

    b. {'wvf human'} - Human shift-invariant optics based on mean wavefront abberration from Thibos et al. (2009, Ophthalmic & Physiological Optics).

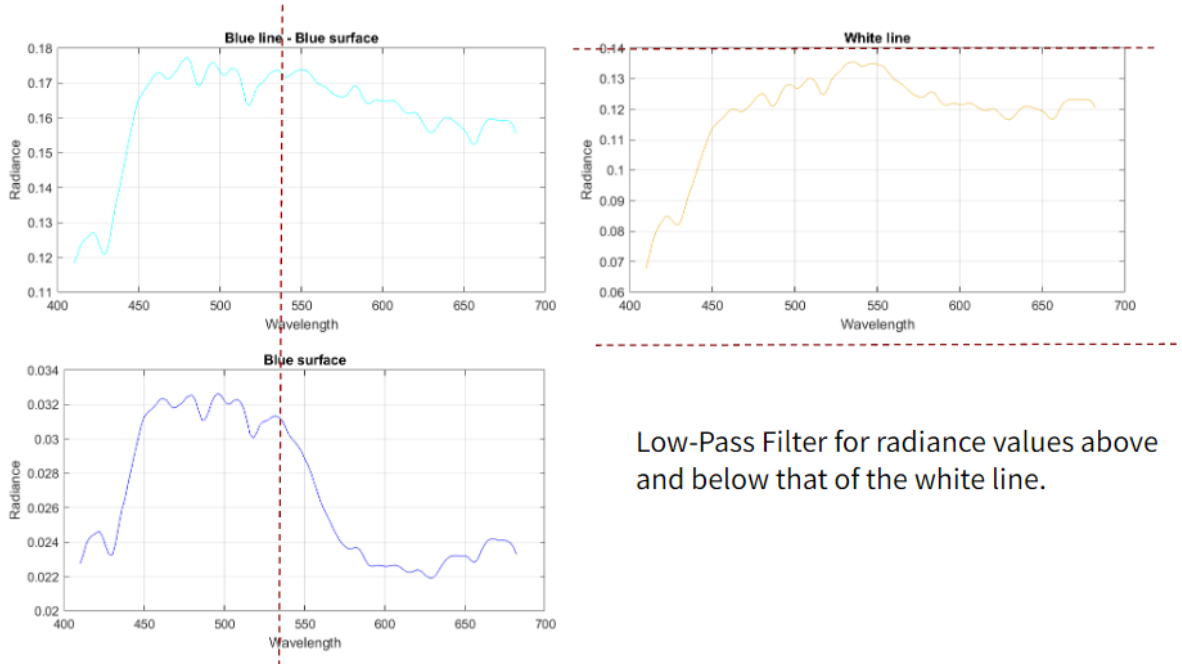
    RGB2XWFormat : Transform an RGB form matrix into an XW (space-wavelength) matrix

    We say matrices in (r, c, w) format are in RGB format. The dimension, w, represents the number of data color bands. The RGB format is useful for imaging. When w = 3, you can use conventional image() routines. When w > 3, use imageSPD. The XW (space-wavelength) format is useful for computation. In this format, for example, XW * spectralFunction yields a spectral response.

    {'wvf human'} - Human shift-invariant optics based on mean wavefront abberration from Thibos et al. (2009, Ophthalmic & Physiological Optics). Optional parameters can be passed for this case.

2. **Initial Approach: Dynamic Filter**

a. Conditional Low-Pass Filter for radiance values above and below that of the white line.



Results

1. **Sun**

Data from the sun was obtained and plotted in terms of the Radiance (fig. 16) and Reflectance (fig. 17). With the reflectance data and spectrums, we tried different mathematical expressions and compared their delta E values. We chose the delta E as our key parameter as it is a standard metric that quantifies and correlates human visual judgement of the difference between 2 colors [5]. The filter for the sun conditions was decided to be the square-root function as it gave the smallest delta E values amongst the other mathematical expressions we tried. A plot of the transmittance through the filter is as shown in fig 15.

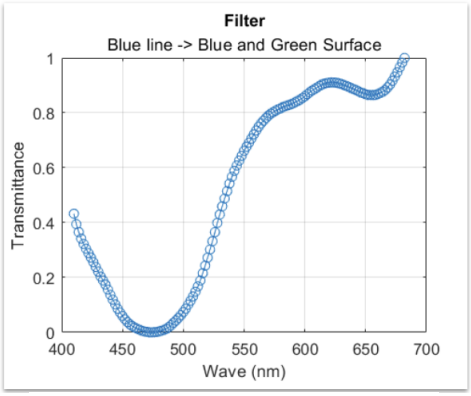


Figure 15: Sun filter created using the square-root function. We decided on square-root after trying different mathematical expressions and comparing their delta E values.

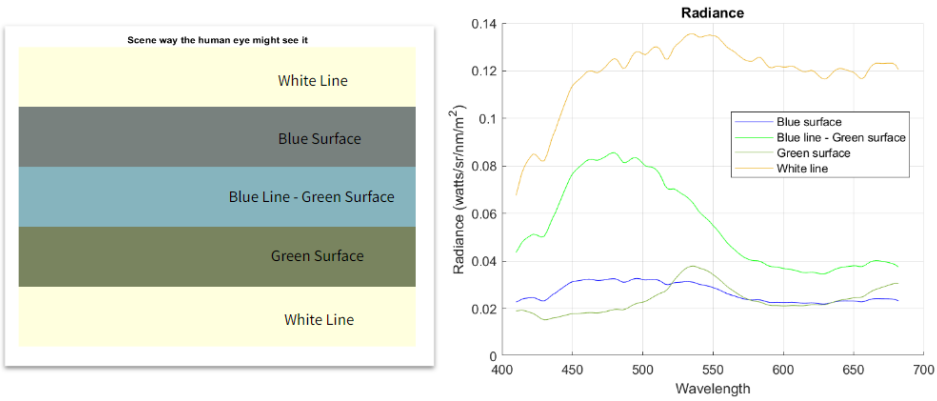


Figure 15: (Left) Color of the scene that the human eye sees. (Right) Plot of radiance against wavelength.

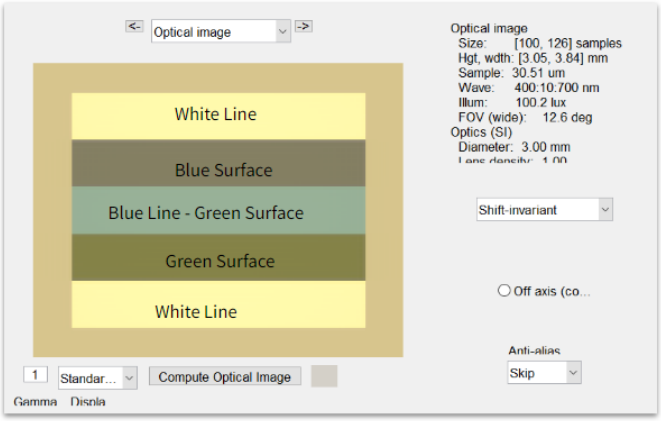


Figure 16: Color of the scene as an optical image.

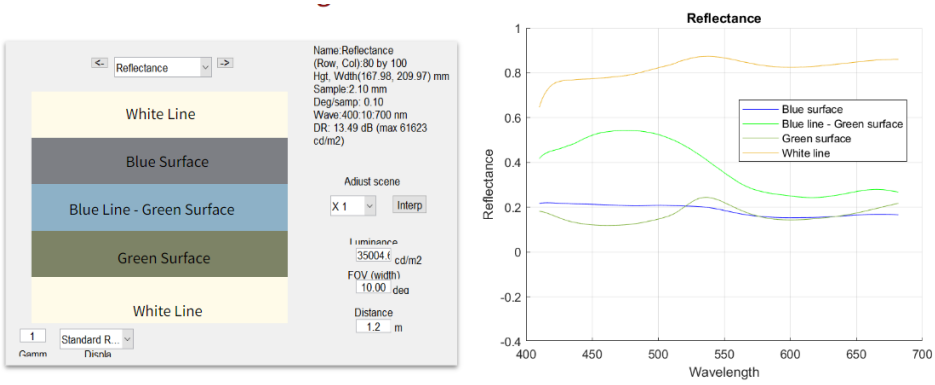


Figure 17: (Left) Color of the scene after taking into account the data from the white calibration square. (Right) Plot of reflectance against wavelength.

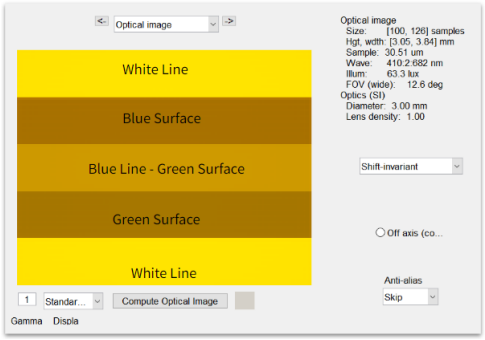


Figure 18: Color of scene after the filter as an optical image.

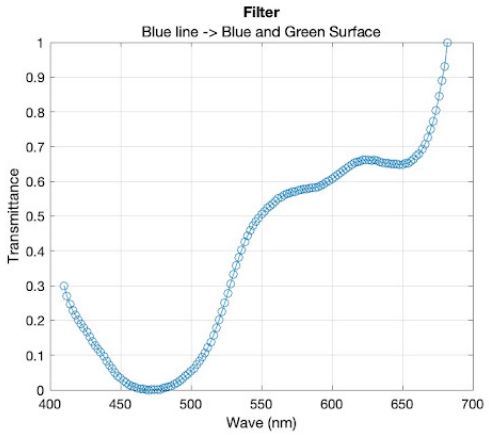
Table 1: Delta E values before and after the filter.

Illuminant	deltaE: Before Filter		deltaE: After Filter	
	Blue line vs. blue surface	Blue line vs. green surface	Blue line vs. blue surface	Blue line vs. green surface
Direct Sun	19.7	18.99	<2	<2

The difference in the values before and after the filter (Table 1) is consistent with the optical image of the scene which seems to show the blue line vs blue surface, and blue line vs green surface, looking more similar/indistinguishable after the filter. A threshold of deltaE below 2 is used because that is the limit at which a human eye is able to distinguish between 2 colors [5]. To better understand the robustness of our filter, we put the data obtained in the shade at the pickleball court through our filter. We also introduced lighting conditions that are similar (D50) and rather different (fluorescent) from the lighting conditions for which our sun filter was created based on.

2. **Shade**

By similar methods as the sunny conditions above,



$$1 - rescale[\mathbf{log10}(BL - GS) + \mathbf{log10}(BL - BS)]$$

Figure 19: Sun filter created using the square-root function. We decided on log-10 after trying different mathematical expressions and comparing their delta E values.

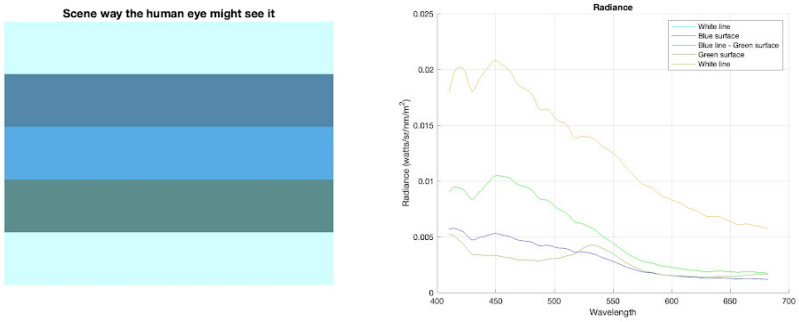


Figure 20: (Left) Color of the scene that the human eye sees. (Right) Plot of radiance against wavelength.

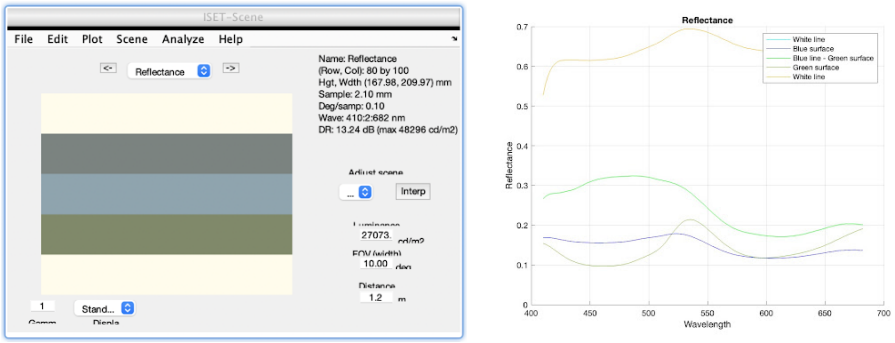


Figure 21: (Left) Color of the scene after taking into account the data from the white calibration square. (Right) Plot of reflectance against wavelength.

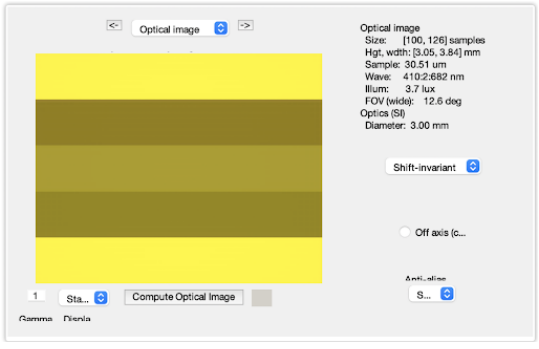


Figure 22: Color of scene after the filter as an optical image.

3. Novel lighting conditions

We subsequently computed the Delta E values to determine if our filter works for the lighting conditions. Out of curiosity, we also created the filter based on data from the different lighting conditions using the mathematical expression square-root as our base. SImilarly, we passed data from the various lighting conditions through the filters and computed the Delta E values. For our filters created using square-root as the starting point, the Delta E values obtained remain below 2 regardless of lighting conditions (Table 2). This suggests our filter is robust and is likely to work for a range of lighting conditions from shade to D50 and fluorescent.

Table 2: Values of Delta E obtained under various lighting conditions and filters.

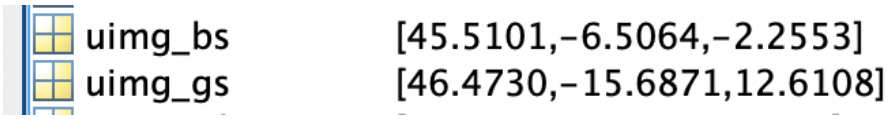
	FilterSun*	FilterFluorescent*	FilterD50*	No Filter
Sun	<2	<2	<2	19.75; 19.00
Shade	<2	<2	<2	17.30; 18.70
Fluorescent	<2	<2	<2	16.84; 21.64
D50	<2	<2	<2	18.25; 20.26

Conclusions

How to Iteratively Create an Appropriate Filter

1. Basics: Inversion, rescaling, summing Green and Blue surface components
    a. Ensure the blue line sufficiently blends in with both.
2. Sun vs. Shade: lower contrast in lines vs. surfaces for latter
    a. Raw reflectance difference must be scaled more severely in shade.
    b. Base 10 Log as core filter function for shade, Sqrt. as that for sun.
3. Shade: use Textile kL=2, NOT screen kL=1
    a. Unfiltered deltaE = 17.3 (realistic, kL=2) vs. 33.04 (kL=1)
        i. for blue line vs. blue surface
4. Cascading filters
    a. log10(log10(x)) → **not** useful in reducing error

Double-checking XYZ values



Post-filter XYZ values of blue and green surface: Different target to achieve

Appendix

Presentation slides

Sign in to access Google Drive Presentation

Code

PSYCH 221 Project - Pickleball Selective Vision Glasses

Priyanka Dilip, Xin-Yi Pan, Maryann Benny Fernandes, Rita Meraz

Using knowledge of color vision, this script generates a filter that enhances and reduces the color perception contrast to address the Pickleball problem. Reflectance data was captured via a spectral radiometer, ISETBIO is used to simulate the lens and filter through which to view the gathered data.

More information: Project Website

Plot radiance of data collected

```
% Set wavelength
wavelength = 410:2:682;

% Read data collected (Sun case)
blue_surface = ieReadSpectra('spd-2022-11-13_blue_surface_in_sun.mat',wavelength);
blue_line_Green = ieReadSpectra('spd-2022-11-13_blue_line_on_green_surface_in_sun.mat',wavelength);
green_surface = ieReadSpectra('spd-2022-11-13_green_surface_in_sun.mat',wavelength);
white_line= ieReadSpectra('spd-2022-11-13_white_line_on_greenblue_surface_in_sun.mat',wavelength);

% Read data collected (Shade case)
% blue_surface = ieReadSpectra('spd-2022-11-13_blue_surface_in_shade.mat',wavelength);
% blue_line_Green = ieReadSpectra('spd-2022-11-13_blue_line_on_green_surface_in_shade.mat',wavelength);
% green_surface = ieReadSpectra('spd-2022-11-13_green_surface_in_shade.mat',wavelength);
% white_line= ieReadSpectra('spd-2022-11-13_white_line_on_greenblue_surface_in_shade.mat',wavelength);

name = ["White line";"Blue surface";"Blue line - Green surface";"Green surface"; "White line"];
radiance_array = [white_line blue_surface blue_line_Green green_surface white_line];
colors = ["c";"b";"g";"#77AC30";"#EDB120"];

% Plot radiance
figure()
for i=1:length(radiance_array(1,:))
    subplot(3,2,i);
    plot(wavelength,radiance_array(:,i),Color=colors(i));
    title(name(i));
    grid("on")
    xlabel('Wavelength');
    ylabel('Radiance (watts/sr/nm/m^2)')
end
```

```
figure()
hold on
for i=2:length(radiance_array(1,:))
    plot(wavelength,radiance_array(:,i),Color=colors(i));
end
hold off
legend(name(2),name(3),name(4),name(5))
title('Radiance');
grid("on")
xlabel('Wavelength');
ylabel('Radiance (watts/sr/nm/m^2)');
```

Create Scenes

```
% Load white calibration data and set it as the illuminant
% SUN
% illuminant_blue_line_Blue = ieReadSpectra('spd-2022-11-13_blue_line_on_blue_surface_in_sun_white_calibration.mat',wavelength);
% illuminant_blue_surface = ieReadSpectra('spd-2022-11-13_blue_surface_in_sun_white_calibration.mat',wavelength);
% illuminant_blue_line_Green = ieReadSpectra('spd-2022-11-13_blue_line_on_green_surface_in_sun_white_calibration.mat',wavelength);
% illuminant_green_surface = ieReadSpectra('spd-2022-11-13_green_surface_in_sun_white_calibration.mat',wavelength);

% All the white calibration measures are similar as expected since these
% data was captured in the sun. An arbitrary white calibration measured was used.
illuminant_white_line = ieReadSpectra('spd-2022-11-13_white_line_on_greenblue_surface_in_sun_white_calibration.mat',wavelength);

% SHADE
% illuminant_blue_surface = ieReadSpectra('spd-2022-11-13_blue_surface_in_shade_white_calibration.mat',wavelength);
% illuminant_blue_line_Green = ieReadSpectra('spd-2022-11-13_blue_line_on_green_surface_in_shade_white_calibration.mat',wavelength);
% illuminant_green_surface = ieReadSpectra('spd-2022-11-13_green_surface_in_shade_white_calibration.mat',wavelength);
% illuminant_white_line = ieReadSpectra('spd-2022-11-13_white_line_on_greenblue_surface_in_shade_white_calibration.mat',wavelength);

% Create a patch of each radiance for easy visualization
patch = zeros(80,100,length(wavelength));
[h,w,lamda] = size(patch);
for r=1:length(wavelength)
    for j=1:length(radiance_array(1,:))
        patch(j*(h/5)-(h/5-1):j*(h/5),:,r) = repmat(radiance_array(r,j),h/5,w);
    end
end
```

```
% Create scene
scene = sceneCreate;
scene = sceneSet(scene,'wave', wavelength);
scene = sceneSet(scene,'name', 'Radiance');
% Set the chosen white calibration data as the illuminant
scene = sceneSet(scene,'illuminantenergy', illuminant_white_line);
% Adjust the current scene illuminant to desired one
% scene = sceneAdjustIlluminant(scene, 'Fluorescent.mat')
scene = sceneSet(scene,'energy', patch);
sceneWindow(scene);

% Preview the scene the way the human eye might see it by rendering it through LMS cone responses
imgXYZ   = sceneGet(scene,'xyz'); % Get 3D array of XYZ values of the data
whiteXYZ = sceneGet(scene,'illuminant xyz'); % Get XYZ values of the illuminant
vcNewGraphWin;
lms =  xyz2lms(imgXYZ, 0, 'Brettel', whiteXYZ); % Transform an XYZ image to Stockman LMS cone format
XYZ = imageLinearTransform(lms, colorTransformMatrix('lms2xyz'));
imagesc(xyz2srgb(XYZ)); %  Convert CIE XYZ to sRGB color space
title('Scene way the human eye might see it')
axis image;
axis off
```

Get reflectance using sceneGet

```
% Get reflectance
reflectance = sceneGet(scene,'reflectance');

% See reflectance
scene_reflectance = scene;
scene_reflectance = sceneSet(scene_reflectance,'name', 'Reflectance');
scene_reflectance = sceneSet(scene_reflectance,'energy', reflectance);
sceneWindow(scene_reflectance);

figure()
hold on
for i=2:length(radiance_array(1,:))
    plot(wavelength,squeeze(reflectance(i*16-11,10,:)),Color=colors(i));
end
hold off
```

```
legend(name(2),name(3),name(4),name(5))
title('Reflectance');
grid("on")
xlabel('Wavelength');
ylabel('Reflectance');
```

Visualize the resulting optical image

```
theOI = oiCreate('wvf human'); % Human shift-invariant optics based on mean wavefront abberration
theOI = oiCompute(theOI, scene);
theOI = oiSet(theOI,'name', 'Optical image');
oiWindow(theOI);
```

Create Filter

This filter reduces the contrast of the blue line for (Pickleball court) and the blue and green backgrounds while keeping a high contrast with the white lines (tennis court). i.e. the blue line, blue background, and green background seem closely the same color while the white line of a different one.

```
%  1:16  blue line
% 17:32  blue surface
% 33:48  blue line on green
% 49:64  green surface
% 65:end white line

filter1 = squeeze(sqrt(reflectance(37,10,:)-reflectance(21,10,:)));
filter2 = squeeze(sqrt(reflectance(37,10,:)-reflectance(53,10,:)));
transmittance = 1-rescale((filter1+filter2));
% save('filterD50', 'transmittance');
% m = matfile('filterFluorescent.mat');
% m = matfile('filterOG.mat'); % Sun filter
% m = matfile('filterD50.mat');
% transmittance = m.transmittance;

ieNewGraphWin;
plot(wavelength,transmittance,'-o');
title('Filter', ...
    'Blue line -> Blue and Green Surface')
grid on;
xlabel('Wave (nm)');
ylabel('Transmittance');
```

```
% Retina image with filter
Photons = oiGet(theOI,'photons');
[Photons, row, col] = RGB2XWFormat(Photons);
Photons = Photons*diag(transmittance);
Photons = XW2RGBFormat(Photons, row, col);

oi = oiSet(theOI,'photons',Photons);
vcAddAndSelectObject(oi);
oiWindow;
vcNewGraphWin;
```

Scene with filter

```
% Preview the scene the way the human eye might see it by rendering it through LMS cone responses
imgXYZ_filter = oiGet(oi,'xyz');
lms =  xyz2lms(imgXYZ_filter(11:end-10,14:end-13,:), 0, 'Brettel', whiteXYZ);
XYZ = imageLinearTransform(lms, colorTransformMatrix('lms2xyz'));
srgb_ = xyz2srgb(XYZ);
imagesc(srgb_);
title('After Filter', ...
    'Way the human eye might see the scene')
axis image;
axis off
```

Error Calculations - DeltaE Calculation

With filter

```
samp_white = squeeze(imgXYZ_filter(1,5+25*1,:)).';
img_white = ieXYZ2LAB(samp_white, double(whiteXYZ));
samp_bs = squeeze(imgXYZ_filter(1,5+25*2,:)).';
img_bs = ieXYZ2LAB(samp_bs, double(whiteXYZ));
samp_gs = squeeze(imgXYZ_filter(1,5+25*3,:)).';
img_gs = ieXYZ2LAB(samp_gs, double(whiteXYZ));
[d_blvsbs, ~] = deltaE2000(img_bs,img_white, [2 1 1] )
[d_blvsgs, ~] = deltaE2000(img_gs,img_white, [2 1 1] )
```

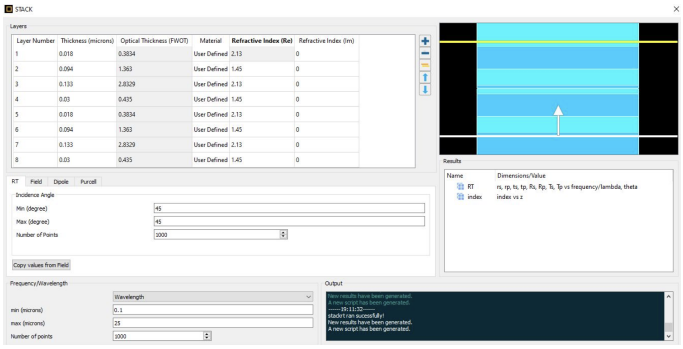
Before filter

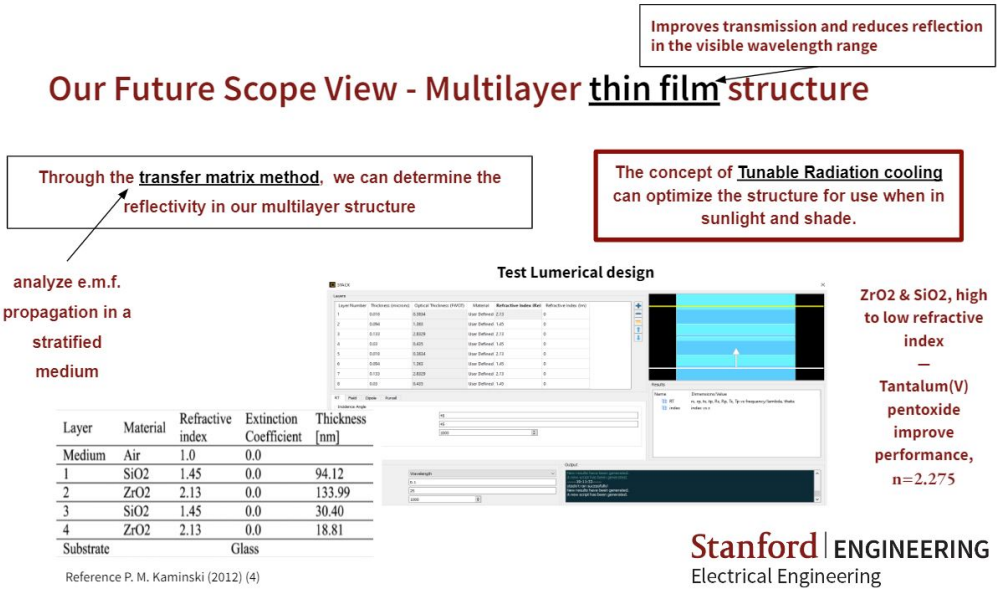
```
unfilt_white = squeeze(imgXYZ(5+16*4,1,:)).';
uimg_white = ieXYZ2LAB(unfilt_white, double(whiteXYZ));
unfilt_bs = squeeze(imgXYZ(5+16*1,1,:)).';
uimg_bs = ieXYZ2LAB(unfilt_bs, double(whiteXYZ));
unfilt_gs = squeeze(imgXYZ(5+16*3,1,:)).';
uimg_gs = ieXYZ2LAB(unfilt_gs, double(whiteXYZ));
[d_orig_blvsbs, ~] = deltaE2000(uimg_bs,uimg_white, [2 1 1] )
[d_orig_blvsgs, ~] = deltaE2000(uimg_gs,uimg_white, [2 1 1] )
```

Our Future Scope View

We have realised that using a multilayer thin film structure that helps improve transmission and reduces reflection in the visible wavelength range could be the optimum solution for building the filter. Using the transfer matrix method that analyses the electromagnetic propagation in a stratified medium could help us determine the reflectivity in the multilayer structure.

Layer	Material	Refractive index	Extinction Coefficient	Thickness [nm]
Medium	Air	1.0	0.0	
1	SiO2	1.45	0.0	94.12
2	ZrO2	2.13	0.0	133.99
3	SiO2	1.45	0.0	30.40
4	ZrO2	2.13	0.0	18.81
Substrate			Glass	





References

1. Wandell, B. A. (1995). Foundations of vision. Sinauer Associates.
2. “The CIEDE2000 Color-Difference Formula: Implementation Notes Supplementary Test Data, and Mathematical Observations,” G. Sharma, W. Wu, E. N. Dalal, submitted to Color Research and Application, January 2004.

3. “Multilayer thin film structures for multifunctional glass: Self-cleaning, antireflective and energy-saving properties,” Corrado Garlisi, et al., April 2015
4. Kaminski, P. M., Lisco, F., & Walls, J. M. (2013). Multilayer broadband antireflective coatings for more efficient thin-film CdTe solar cells. IEEE Journal of Photovoltaics, 4(1), 452-456
5. View Sonic, https://www.viewsonic.com/eu/colorpro/articles/detail/deltae2color-accuracy_3

Thank you!