Color Metamers Notebook

Section #1: Introduction

In this notebook, we will explore some of the math involved in the color and perception of color. Specifically, we will look at the concept of color metamerism--how two or more colors can be perceived to be the same color, despite being fundamentally different in their composition.

Color is a phenomenon particularly associated with humans, but other animals do see color as well. The reason color exists is because of light and special cells in our eyes called “cones” that are sensitive to light. Light itself is an electromagnetic wave and fundamentally defined by wavelength (λ), and human cone cells are sensitive to a subset of the Electromagnetic Spectrum called the Visible Light Spectrum, which roughly includes λ values from 380 to 780 nanometers.

Usually, light that enters your eye is polychromatic, meaning it composed of various amounts and values of λ. From there, the three types of cone cells in your eye respond by sending a signal to your brain. This signal is ultimately what your brain labels as a color.

Section 2: Color Things

The system of color perception is three-fold [1]. Below is a brief description of each:

1. **Type of Light**

   Light is often polychromatic, meaning it is composed of multiple wavelengths of light. We can ignore the composition of wavelengths outside the Visible Spectrum because they typically do not interfere with the rest of the system. It is helpful to think of light as a spectral power distribution (SPD), which we will define as the function $L(\lambda)$.

2. **How Objects Reflect Light**

   The reason we see objects of a certain color is because light bounces from those objects. However, the incident light (or SPD) might not be the same. This is because an object might reflect more or less light of a certain wavelength than another. We can think of the SPD that bounces off the object as the following:

   $$L_r(\lambda) = R(\lambda) * L_i(\lambda),$$

   where $L_r(\lambda)$ is the SPD that is reflected from the object, R(λ) is the reflectance distribution of the object, and $L_i(\lambda)$ is the incident SPD.

   For our purposes, we will assume $L(\lambda) = L_r(\lambda) = L_i(\lambda)$.

3. **How Observer Interprets Light**

   Inside your eye, there are three cone called L, M, and S cones. They are all sensitive to light at different but overlapping wavelengths. How your eye interprets the light it sees into color is a complicated process. One model that mathematically explains color was developed by the CIE (Commission Internationale de l'éclairage).

In order to use the CIE colorimetric system, we must understand that any color can be represented as a combination of different powers of red, green, and blue light. The three primaries used in this system are single wavelengths of 700nm for red, 546.1nm for green, and 435.8nm for blue light [2]. The unit intensities are proportional, meaning if 1 $\frac{\text{cd}}{m^2}$ of red light was used, then 4.5907 $\frac{\text{cd}}{m^2}$ of green and 0.0601 $\frac{\text{cd}}{m^2}$ of blue was used.

In 1931, the CIE published a set of functions called the RGB Color Matching Functions (CMFs). These functions were determined based on experimental observations by W. D. Wright and J. Guild [2]. Essentially, they tried to determine the amount of each primary needed to match a single wavelength across a range of wavelengths for human vision. Because the range of these functions were negative, the CIE published a transformed version called the XYZ CMFs. These were positive for all values.

The XYZ CMFs are as follows:

$$X = \int_{380}^{780} L(\lambda)\cdot\overline{x}(\lambda)\,d\lambda$$
$$Y = \int_{380}^{780} L(\lambda)\cdot\overline{y}(\lambda)\,d\lambda$$
$$Z = \int_{380}^{780} L(\lambda)\cdot\overline{z}(\lambda)\,d\lambda$$

where the set $\{X, Y, Z\}$ is known as the tristimulus value, and $\overline{x}(\lambda)$, $\overline{y}(\lambda)$, and $\overline{z}(\lambda)$ are the CMFs. They represent a standard observer, as defined by the CIE. Furthermore, the tristimulus value is the amount for each primary needed to achieve a match for a given set of CMFs and SPD [3]. For our purposes, we can think of a tristimulus value representing a color, and thus every color being represented as three numbers.

It is possible that one SPD yields the same tristimulus value as another. For example, a SPD representing yellow monochromatic light will yield the same tristimulus value as a SPD of particular mixture of green and red monochromatic lights. This can be observed by combing green and red together, which forms yellow. In this context, the yellow monochromatic light and the combination of red and green monochromatic lights are metameric, as they both yield the same response.

What we want to know is given a target SPD (a particular color), what combinations of other proposed SPDs (other colors) would produce the same tristimulus value? This is essentially asking what proposed SPDs are metameric to the target SPD, or what combinations of colors are metameric to a particular color.

Section 3: Linear Algebra

Vector Space of SPDs

Let $\mathbb{U}$ be the vector space of all SPDs. Mathematically:

$$\mathbb{V} = \{L\ s.t.: L \in [380,\ 780]\} \subset \mathbb{U},$$

where L is any SPD function that is nonnegative. We will use $\mathbb{V}$ as it is the vector space relevant for human color perception[1]. The space of $\mathbb{V}$ is very large, and can be thought of as "infinite."

[1]In actuality, $\mathbb{V}$ is not a vector space because there is no such thing as a negative SPD. For this notebook, we will pretend it is.

Color Perception as a Linear Transformation

Color perception can be thought of as a transformation $T : \mathbb{V} \to \mathbb{R}^3$ defined as the following:

$$T(L(\lambda)) = \int_{380}^{780} L(\lambda) \cdot \text{CMFs}(\lambda)\, d\lambda = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix},$$

where

$$\text{CMFs}(\lambda) = \begin{pmatrix} \overline{x}(\lambda) \\ \overline{y}(\lambda) \\ \overline{z}(\lambda) \end{pmatrix},$$

and

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

is the vector of the tristimulus value. This is a linear transformation as integration is linear.

Injectivity

T is not injective because we know metamers exist. This means $\exists\, L_1, L_2$, where $L_1(\lambda) \neq L_2(\lambda)$ such that $T(L_1(\lambda)) = T(L_2(\lambda))$. Because the space of $\mathbb{V}$ is so large, there are an “infinite” number of L_1 SPD’s that are metameric to one L_2 SPD.

To provide an analogy, suppose you went shopping in a store and had a budget, b. There are virtually an infinite amount of combinations of items you can purchase that exactly meet you budget, as long as the store has an infinite amount of items and infinitesimal price variation you can choose from.

Section #4: CIE Equations

CIE XYZ CMFs

There exists analytical approximations to the 1931 CIE $\overline{XYZ}$ CMFs [4]. They are defined below:

In[∘]:=
```
listOne = {0.362, 1.056, -0.065, 0.821, 0.286, 1.217, 0.681};
listTwo = {442.0, 599.8, 501.1, 568.8, 530.9, 437.0, 459.0};
listThree = {0.0624, 0.0264, 0.0490, 0.0213, 0.0613, 0.0845, 0.0385};
listFour = {0.0374, 0.0323, 0.0382, 0.0247, 0.0322, 0.0278, 0.0725};

tableConstants = TableForm[
    {listOne, listTwo, listThree, listFour},
    TableHeadings→{{"α", "β", "γ", "δ"}, {"x0", "x1", "x2", "y0", "y1", "z0", "z1"}}
];

H[x_] := UnitStep[x];
S[x_, y_, z_] := y(1 - H[x]) + z * H[x];

α = 1;
β = 2;
γ = 3;
δ = 4;

getItemx[col_, i_] := tableConstants⟦1, col⟧⟦i⟧;
getItemy[col_, i_] := tableConstants⟦1, col⟧⟦i + 3⟧;
getItemz[col_, i_] := tableConstants⟦1, col⟧⟦i + 5⟧;
```

$\tilde{x}_{31}[\lambda\_]$:= $\sum_{i=1}^{3}$getItem$_x$[α, i] * Exp$\left[-\frac{1}{2}\right.$((λ - getItem$_x$[β, i]) * S[λ - getItem$_x$[β, i], getItem$_x$

$\tilde{y}_{31}[\lambda\_]$:= $\sum_{i=1}^{2}$getItem$_y$[α, i] * Exp$\left[-\frac{1}{2}\right.$((λ - getItem$_y$[β, i]) * S[λ - getItem$_y$[β, i], getItem$_y$

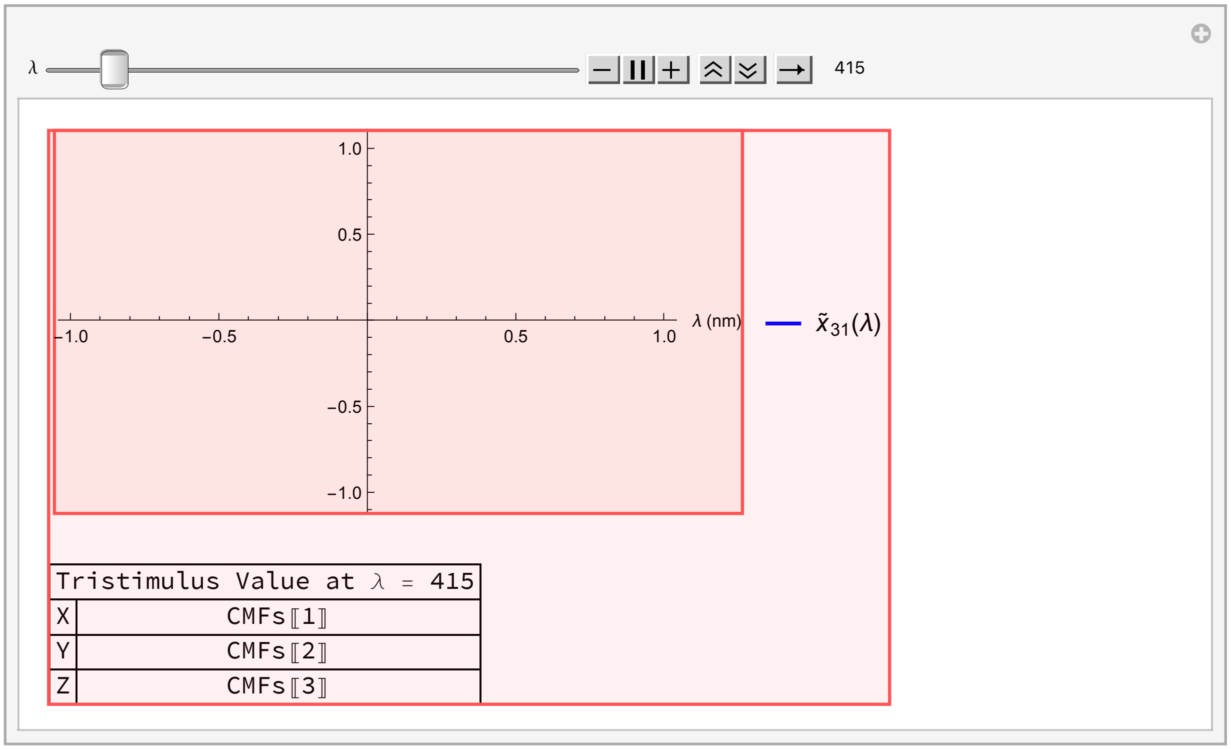
$\tilde{z}_{31}[\lambda\_]$:= $\sum_{i=1}^{2}$getItem$_z$[α, i] * Exp$\left[-\frac{1}{2}\right.$((λ - getItem$_z$[β, i]) * S[λ - getItem$_z$[β, i], getItem$_z$

CMFs = {$\tilde{x}_{31}$[λ], $\tilde{y}_{31}$[λ], $\tilde{z}_{31}$[λ]};

Below is a widget you can play to how the tristimulus value depends on the three functions:

In[ ]:=
```
Manipulate[
    Row[
        {
            Show[
                Plot[
                    CMFs, {λ, 380, 780},
                    PlotStyle→{Red, Green, Blue},
                    PlotLegends→{"x̃31(λ)", "ỹ31(λ)", "z̃31(λ)"},
                    AxesLabel→{"λ (nm)"},
                    PlotRange→All,
                    ImageSize→Medium
                ],
                Graphics[
                    {
                        PointSize[0.02],
                        Point[{currentValue, #}] & /@ CMFs /. λ→currentValue
                    }
                ]
            ],
            Module[
                {values},
                values = CMFs /. λ → currentValue;
                Grid[
                    {
                        {StringForm["Tristimulus Value at λ = ``", currentValue], SpanFro
                        {"X", values⟦1⟧},
                        {"Y", values⟦2⟧},
                        {"Z", values⟦3⟧}
                    },
                    Frame → All
                ]
            ]
        }
    ],
    {
        {currentValue, 380, "λ"},
        380, 780, 1,
        Appearance → "Labeled"
    },
    ControlType → Animator
]
```

Out[ ]=



Tristimulus Value at λ = 415	
X	CMFs⟦1⟧
Y	CMFs⟦2⟧
Z	CMFs⟦3⟧

Section #5: Red + Green = Yellow

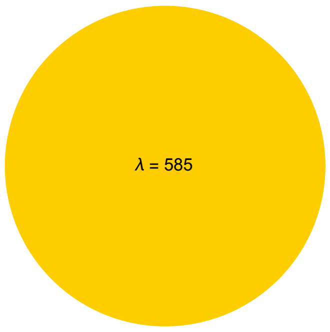
Confirming Red + Green = Yellow

One of the most common examples of metamers is that yellow spectral light is perceived the same as a combination of green and red spectral light. In the example below, we picked values of λ for red, green, and yellow light. We plotted each color individually as well as the combination of green and red[1]. We assume there is an equal amount of each light. To determine if there is a metamer, we used the dot product approach and a small angle of 3° as our threshold. The approach is documented in § 6:

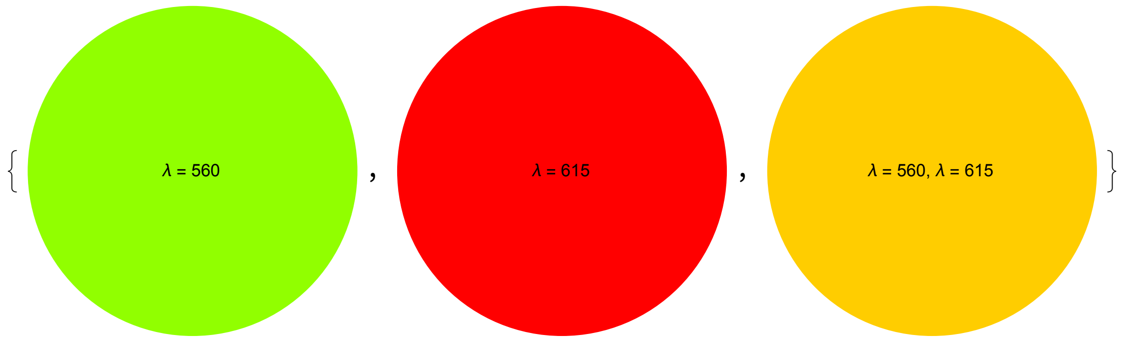
In[ ]:=
```
(*We are assumming there is an equal amount of yellow, green, and red.
Essentially, an absolute SPD at the defined values of λ*)
yellow = CMFs /. λ→585;
green = CMFs /. λ→560;
red = CMFs /. λ→615;



degreeBetween[targetLight_, otherLights_] := ArcCos[targetLight.Total[otherLights] / (Norm[targetLight] * Norm[Total[otherL

StringForm["Angle between yellow and (red + green) tristimulus vectors: ``°", degreeBetwee

Graphics[
    {
    XYZColor[yellow], Disk[], Black, Text["λ = 585"]
    }
]

{
    Graphics[
        {
            XYZColor[green], Disk[], Black, Text["λ = 560"]
        }
    ],
    Graphics[
        {
            XYZColor[red], Disk[], Black, Text["λ = 615"]
        }
    ],
    Graphics[
        {
            Blend[{XYZColor[green], XYZColor[red]}], Disk[], Black, Text["λ = 560, λ = 61
        }
    ]
}
```

Out[ ]=
```
Angle between yellow and (red + green) tristimulus vectors: 2.83267°
```

Out[ ]=



Out[ ]=



Section #6: Calculating Monochromatic Metamers

Methodology

Mathematically, we know metamers exist and there are infinite number of them. Computationally, we are interested in what the specific metamers are. Below is the process we used for calculating monochromatic metamers (of monochromatic light):

1. Defining SPD and Linear Transformation
2. Selecting Target SPD
3. Defining Proposed SPDs and Their Tristimulus Values
4. Determining If Proposed SPDs are Metameric to Target SPD

Defining SPD and Linear Transformation

First, we need to define our linear transformation:

In[•]:=
```
(*The left and right bounds of integration.*)
leftBound = 380;
rightBound = 780;

(*Defining the linear tranformation from SPD to tristimulus value.*)
T[SPD_] := Integrate[CMFs * SPD, {λ, leftBound, rightBound}];
```

Next, we will define our monochromatic SPD:

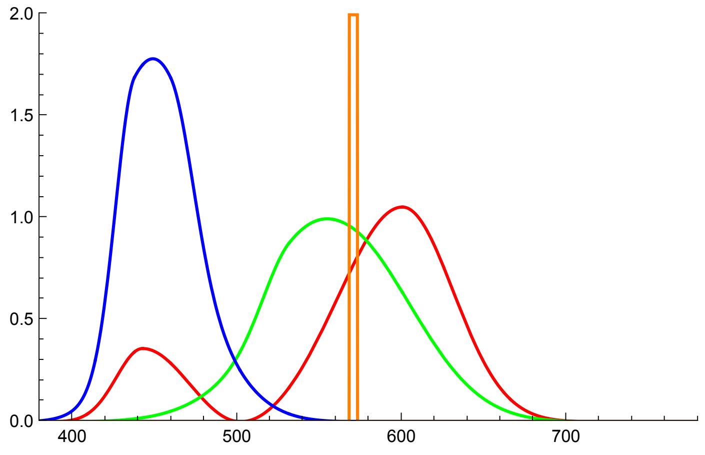
In[•]:=
```
(*Defining the monochromatic SPD function and its width and height.*)
width = 10;
height = 2;

monoSPD[x_, center_] := height * Exp[-(x - center)^2 / (2 * width^2)]; (*Weird results wit

                                   λ - center
monoSPD[λ_, center_] := height * UnitBox[ ---------- ];
                                     width
                                     -----
                                       2

Off[General::munfl] (*Very small values of e^-value present a warning about precision being
```

Selecting Target SPD

Since we are ultimately referring to the tristimulus value of any SPD, we will also define that value as we refer to it often. We chose our target SPD to be at 570nm. We also plotted it in comparison to the CMFs below:

In[ ]:=
```
(*Defining the target SPD to which we want to find a linear combinaton for.*)
p₀ := 570;
targetSPD = monoSPD[λ, p₀];
targetTristimulusValues = T[targetSPD];


plot1 = Plot[CMFs, {λ, leftBound, rightBound}, PlotStyle→{Red, Green, Blue}, PlotRange→{{
plot2 = Plot[monoSPD[λ, p₀], {λ, leftBound, rightBound}, PlotStyle→Orange, Exclusions→Non
Show[plot1, plot2]
```

Out[ ]=



Defining Proposed SPDs and Their Tristimulus Values

As there are infinite number of SPDs that are metameric to our target SPD, we need to define the total number of possible proposed SPDs we will evaluate. This will require defining the step and the number of linear combinations (proposed SPDs) we will consider at once:

In[ ]:=
```
(*400 divided by step is the number of integrals needed to be solved.*)
step = 10;

(*Number of proposed SPDs we want to consider at once.*)
n = 2;

(*All values of λ where we will consider a proposed SPD.*)
proposedSPDRange = Range[leftBound, rightBound, step];

(*The Cartesian Product of the n proposed SPDs.*)
monoLights = DeleteDuplicatesBy[Tuples[proposedSPDRange, n], Sort];
```

Now, we can calculate all the tristimulus values for every possible combination (defined above) beforehand:

```
In[ ]:= (*Mapping all possible proposed SPDs at λ with their corresponding tristimulus values.*)
mappingFunction[i_] := T[monoSPD[λ, i]];
mappings = AssociationMap[mappingFunction, proposedSPDRange];
```

Determining If Proposed SPDs are Metameric to Target SPD

There are two approaches in determining if the set of proposed SPDs is metameric to the target SPD. They are detailed below:

Approach #1: Exact Solutions

In this approach, we will utilize the linear system

$$A\vec{x} = \vec{b},$$

where

$$A = (\, T(L_1(\lambda))\,|\,T(L_2(\lambda))\,|\cdots|\,T(L_n(\lambda))\,),$$

where $L_1,\ L_2,\ \ldots,\ L_n$ represent the n proposed SPDs and

$$\vec{x} = \begin{pmatrix} k_0 \\ k_1 \\ k_2 \end{pmatrix},$$

where k_0, k_1, and k_2 are the coefficients for the proposed SPDs and

$$\vec{b} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix},$$

where X, Y, and Z are the tristimulus values of the target SPD.

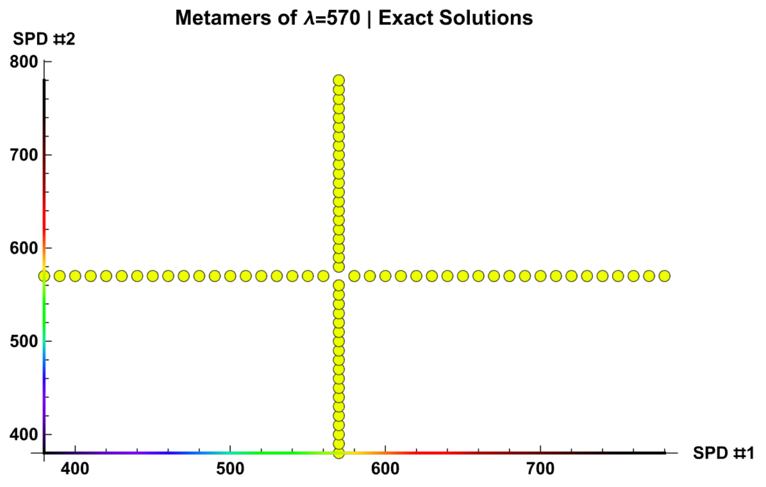
If there exists solutions for the system, then that means the set of n proposed SPDs are metameric to the target SPD. The coefficients of the system are the intensities of each proposed SPD necessary to produce a metamer to the target SPD. Below is the algorithm utilized:

```
Print["Trials: ", Length[monoLights]];
b = targetTristimulusValues;

metamers = ParallelTable[
    (*Get the tristimulus values for the current set of proposed SPDs.*)
    proposedTristimulusValues = Map[mappings, monoLights⟦i⟧];

    (*Convert so that columns become the tristimulus values.*)
    A = Transpose[proposedTristimulusValues];

    x⃗ = Table[Symbol["k" <> ToString@i], {i, 1, n}];

    solution = FindInstance[A.x⃗ == b && AllTrue[x⃗, NonNegative], x⃗, WorkingPrecision→10];

    If[
        Length[solution] > 0,
        AssociationThread[monoLights⟦i⟧, Values[solution⟦1⟧]],
    ],
    {i, 1, Length[monoLights]}
];

(*Only select where metamers are found.*)
metamers = Select[metamers, AssociationQ[#] && Length[Keys[#]] == n &];
Print[StringForm["Pair of `` Monochromatic SPDs Metameric to λ=``: ``", n, p₀, Length[meta
```

```
Trials: 861

Pair of 2 Monochromatic SPDs Metameric to λ=570: 40
```

```
fullList = Join[metamers, Map[Reverse[#] &, metamers]]; (*Accounting for removed duplicat

colors = Map[ColorData["VisibleSpectrum"], Keys[fullList], {2}]; (*List of two-tuple colo

markers = PieChart[fullList[[#]], ChartStyle→colors[[#]], ImageSize→7] & /@ Range[Length[full

ListPlot[
 {#} & /@ Keys[fullList],
 PlotLabel→StringForm["Metamers of λ=`` | Exact Solutions", p₀],
 AxesLabel→{"SPD #1", "SPD #2"},
 PlotMarkers→markers,
 LabelStyle→Directive[Bold, Black],
 Epilog→Style[{
    AbsoluteThickness@1.5,
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{380, x}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ],
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{x, 380}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ]
    }, Antialiasing→False
   ],
 AxesOrigin→{380, 380}
]
```

Out[•]=



Approach #2: Dot Product

In this approach, we utilize the following definition of the dot product of two vectors $\vec{u}$, $\vec{v}$:

$$\vec{u} \cdot \vec{v} = \|\vec{u}\| \, \|\vec{v}\| \cos\theta,$$

where $\vec{u}$ is the tristimulus value of the target SPD and

$$\vec{v} = A\vec{x},$$

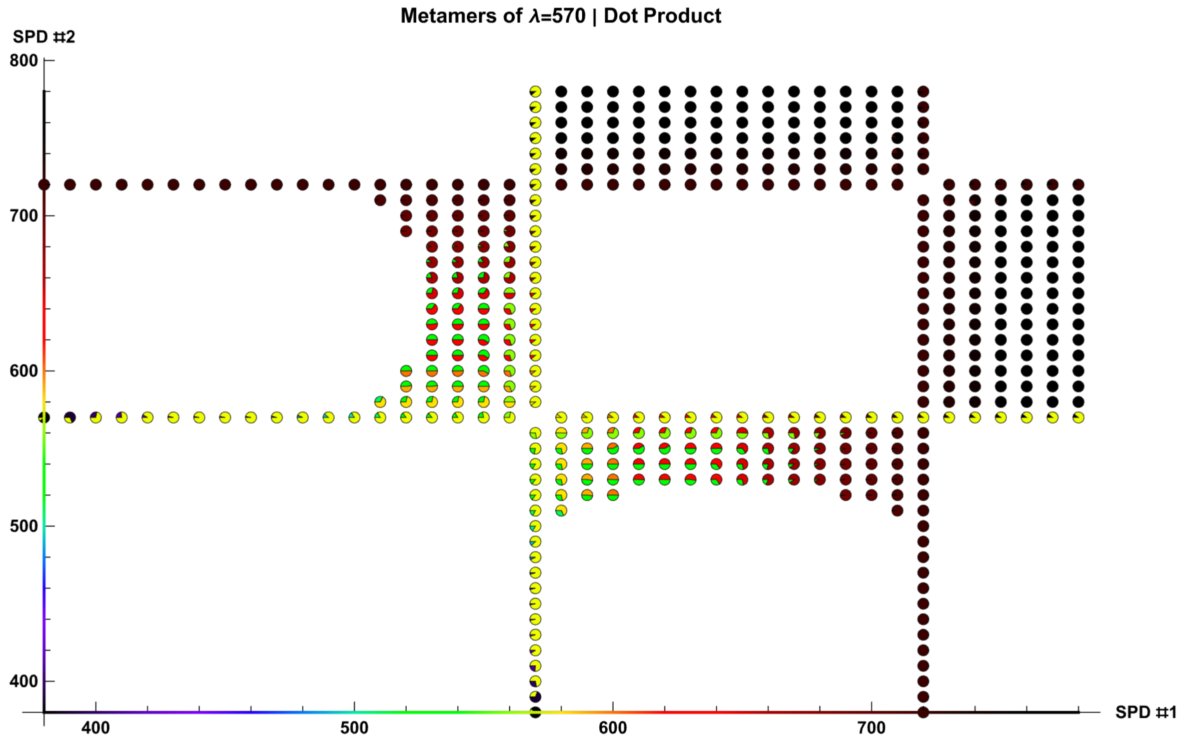
is the dot product of the tristimulus values of the proposed SPDs and coefficients (see Approach #1) and θ represents the angle $\vec{u}$ and $\vec{v}$.

By solving for θ being smaller than some value where there are solutions for $\vec{x}$, we consider a larger array of solutions. The closer (in hues) θ is to 0°, the more metameric the combination of the proposed SPDs are. With this approach, we can obtain approximate metamers, which are more applicable in the real world than the exact ones by the first approach. Below is the algorithm utilized:

```
In[ ]:= (*Get the angle in degrees between the two vectors.*)

                                          u.v              180
        degreeBetween[u_, v_] := ArcCos[―――――――――――――――――] * ―――;
                                       Norm[u] * Norm[v]     Pi

        (*Set a threshold.*)
        degreeMax = 3;

        Print["Trials: ", Length[monoLights]];
        u := targetTristimulusValues;

        metamers = ParallelTable[
            (*Gets the tristimulus values for every light in monoLights.*)
            proposedTristimulusValues = Map[mappings, monoLights⟦i⟧];

            (*Convert so that columns become the tristimulus values.*)
            A = Transpose[proposedTristimulusValues];

            x⃗ = Table[Symbol["k" <> ToString@i], {i, 1, n}];

            v = A.x⃗;

            (*Get degrees between the two vectors.*)
            degree = degreeBetween[u, v];
            solution = FindInstance[degree ≤ degreeMax && AllTrue[x⃗, NonNegative], x⃗, WorkingPrec

            If[
                Length[solution] > 0,
                AssociationThread[monoLights⟦i⟧, Values[solution⟦1⟧]],
            ],
            {i, 1, Length[monoLights]}
        ];

        (*Only select where metamers are found.*)
        metamers = Select[metamers, AssociationQ[#] && Length[Keys[#]] == n &];
        Print[StringForm["Pair of `` Monochromatic SPDs Metameric to λ=``: ``", n, p₀, Length[meta

        Trials: 861

        Pair of 2 Monochromatic SPDs Metameric to λ=570: 227
```

In[ ]:=
```
fullList = Join[metamers, Map[Reverse[#] &, metamers]]; (*Accounting for removed duplicat

colors = Map[ColorData["VisibleSpectrum"], Keys[fullList], {2}]; (*List of two-tuple colo

markers = PieChart[fullList⟦#⟧, ChartStyle→colors⟦#⟧, ImageSize→7] & /@ Range[Length[full

ListPlot[
 {#} & /@ Keys[fullList],
 PlotLabel→StringForm["Metamers of λ=`` | Dot Product", p₀],
 AxesLabel→{"SPD #1", "SPD #2"},
 PlotMarkers→markers,
 LabelStyle→Directive[Bold, Black],
 Epilog→Style[{
    AbsoluteThickness@1.5,
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{380, x}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ],
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{x, 380}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ]
    }, Antialiasing→False
   ],
 AxesOrigin→{380, 380}
]
```

Out[∘]=



3D Example

We tried the above for 2 SPDs, now we will try for 3 SPDs. Big thank you to the StackOverflow responses that helped me figure out some visualization:

In[∘]:=
```
(*400 divided by step is the number of integrals needed to be solved.*)
step = 10;

(*Number of proposed SPDs we want to consider at once.*)
n = 3;

(*All values of λ where we will consider a proposed SPD.*)
proposedSPDRange = Range[leftBound, rightBound, step];

(*The Cartesian Product of the n proposed SPDs.*)
monoLights = DeleteDuplicatesBy[Tuples[proposedSPDRange, n], Sort];
```

In[∘]:=
```
(*Mapping all possible proposed SPDs at λ with their corresponding tristimulus values.*)
mappingFunction[i_] := T[monoSPD[λ, i]];
mappings = AssociationMap[mappingFunction, proposedSPDRange];
```

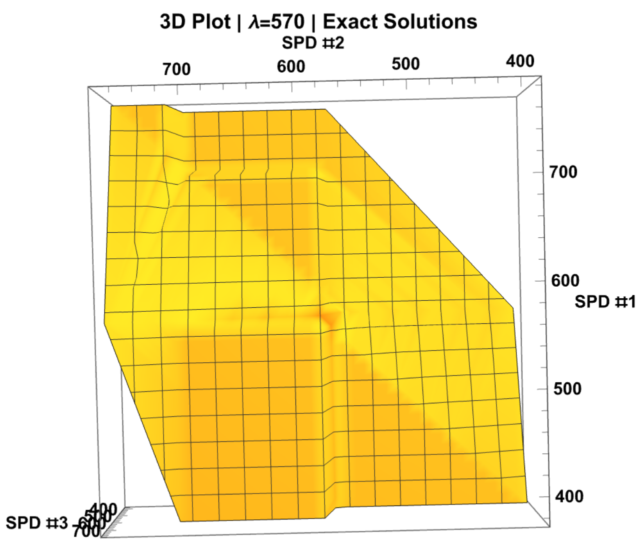
Approach # 1

In[ ]:=
```
Print["Trials: ", Length[monoLights]];
b = targetTristimulusValues;

metamers = ParallelTable[
    (*Get the tristimulus values for the current set of proposed SPDs.*)
    proposedTristimulusValues = Map[mappings, monoLights⟦i⟧];

    (*Convert so that columns become the tristimulus values.*)
    A = Transpose[proposedTristimulusValues];

    x⃗ = Table[Symbol["k" <> ToString@i], {i, 1, n}];

    solution = FindInstance[A.x⃗ == b && AllTrue[x⃗, NonNegative], x⃗, WorkingPrecision→10];

    If[
        Length[solution] > 0,
        AssociationThread[monoLights⟦i⟧, Values[solution⟦1⟧]],
    ],
    {i, 1, Length[monoLights]}
];

(*Only select where metamers are found.*)
metamers = Select[metamers, AssociationQ[#] && Length[Keys[#]] == n &];
Print[StringForm["Pair of `` Monochromatic SPDs Metameric to λ=``: ``", n, p₀, Length[meta
```

```
Pair of 3 Monochromatic SPDs Metameric to λ=570: 2642
```

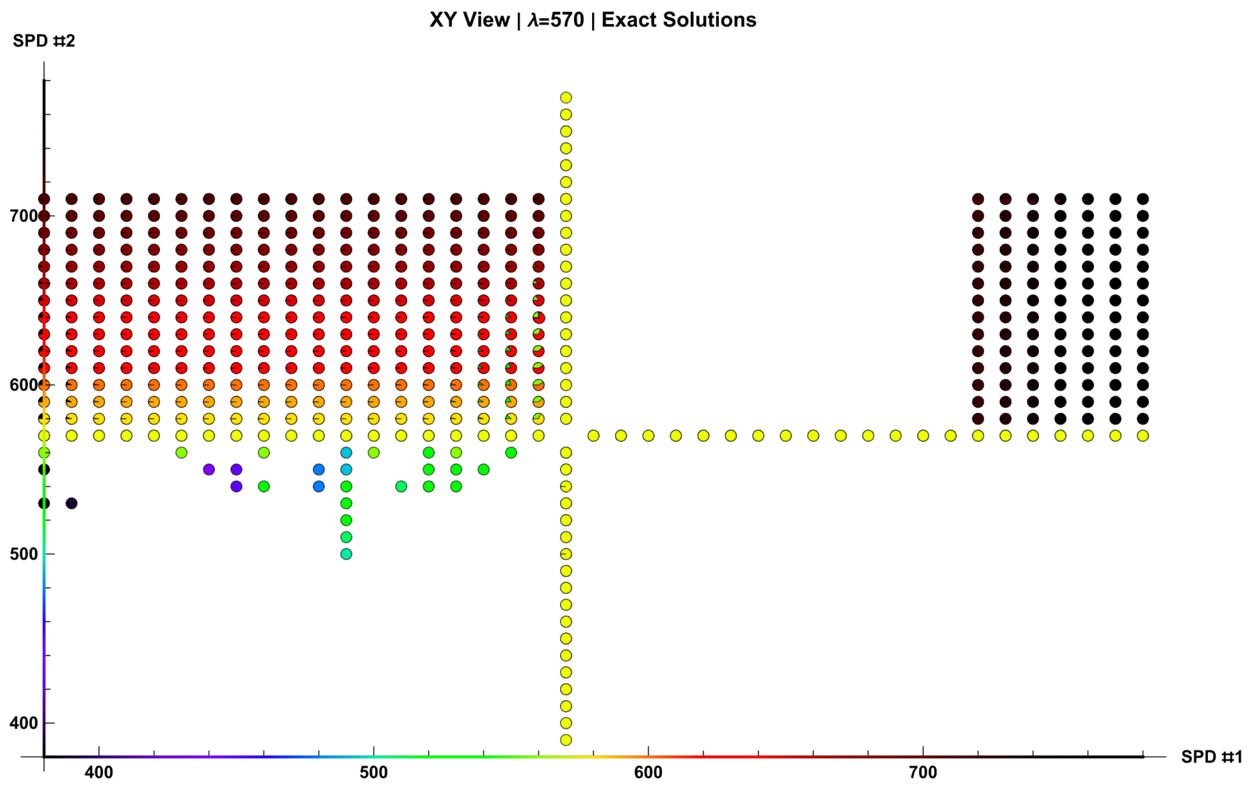
In[ ]:=
```
fullList = Join[metamers, Map[Reverse[#] &, metamers]]; (*Accounting for removed duplicat

ListPlot3D[
 Keys[fullList],
 PlotLabel→StringForm["3D Plot | λ=`` | Exact Solutions", p₀],
 AxesLabel→{"SPD ♯1", "SPD ♯2", "SPD ♯3"},
 LabelStyle→Directive[Bold, Black],
 AxesOrigin → {380, 380}
]
```

Out[ ]=

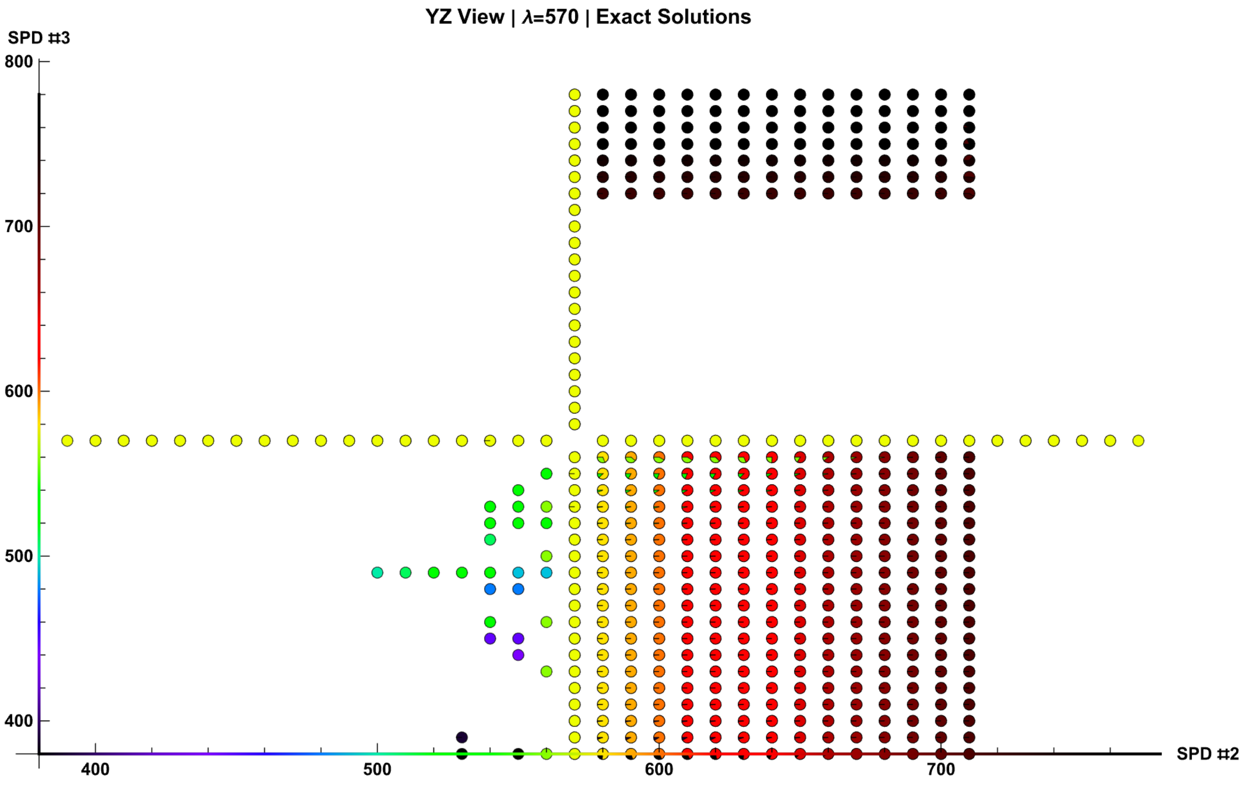


In[ ]:=
```
subsets = Table[
    AssociationThread[
        #,
        Lookup[fullList⟦i⟧, #]] & /@ Subsets[Keys[fullList⟦i⟧], {2}],
    {i, Length[fullList]}
];

components = Transpose[subsets];

XY = components⟦1⟧;
colorsXY = Map[ColorData["VisibleSpectrum"], Keys[XY], {2}]; (*List of two-tuple colors b
markersXY = PieChart[XY⟦#⟧, ChartStyle→colorsXY⟦#⟧, ImageSize→7] & /@ Range[Length[XY]];

XZ = components⟦2⟧;
colorsXZ = Map[ColorData["VisibleSpectrum"], Keys[XZ], {2}]; (*List of two-tuple colors b
markersXZ = PieChart[XZ⟦#⟧, ChartStyle→colorsXZ⟦#⟧, ImageSize→7] & /@ Range[Length[XZ]];

YZ = components⟦3⟧;
colorsYZ = Map[ColorData["VisibleSpectrum"], Keys[YZ], {2}]; (*List of two-tuple colors b
markersYZ = PieChart[YZ⟦#⟧, ChartStyle→colorsYZ⟦#⟧, ImageSize→7] & /@ Range[Length[YZ]];

ListPlot[
 {#} & /@ Keys[XY],
 PlotLabel→StringForm["XY View | λ=`` | Exact Solutions", p₀],
 AxesLabel→{"SPD #1", "SPD #2"},
 PlotMarkers→markersXY,
 LabelStyle→Directive[Bold, Black],
 Epilog → Style[{
    AbsoluteThickness@1.5,
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{380, x}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ],
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{x, 380}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ]
    }, Antialiasing → False
   ],
 AxesOrigin → {380, 380}
]

ListPlot[
 {#} & /@ Keys[YZ],
 PlotLabel→StringForm["YZ View | λ=`` | Exact Solutions", p₀],
 AxesLabel→{"SPD #2", "SPD #3"},
 PlotMarkers→markersYZ,
 LabelStyle→Directive[Bold, Black],
 Epilog → Style[{
    AbsoluteThickness@1.5,
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{380, x}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ],
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{x, 380}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ]
    }, Antialiasing → False
   ],
 AxesOrigin → {380, 380}
]

ListPlot[
 {#} & /@ Keys[XZ],
 PlotLabel→StringForm["XZ View | λ=`` | Exact Solutions", p₀],
  AxesLabel→{"SPD #1", "SPD #3"},
  PlotMarkers→markersXZ,
  LabelStyle→Directive[Bold, Black],
  Epilog → Style[{
     AbsoluteThickness@1.5,
     Line[#, VertexColors → #2] & @@ Transpose@Table[
        {{380, x}, ColorData["VisibleSpectrum"][x]}, {x,
         Subdivide[380, 780, 20]}
        ],
     Line[#, VertexColors → #2] & @@ Transpose@Table[
        {{x, 380}, ColorData["VisibleSpectrum"][x]}, {x,
         Subdivide[380, 780, 20]}
        ]
     }, Antialiasing → False
    ],
  AxesOrigin → {380, 380}
 ]
```

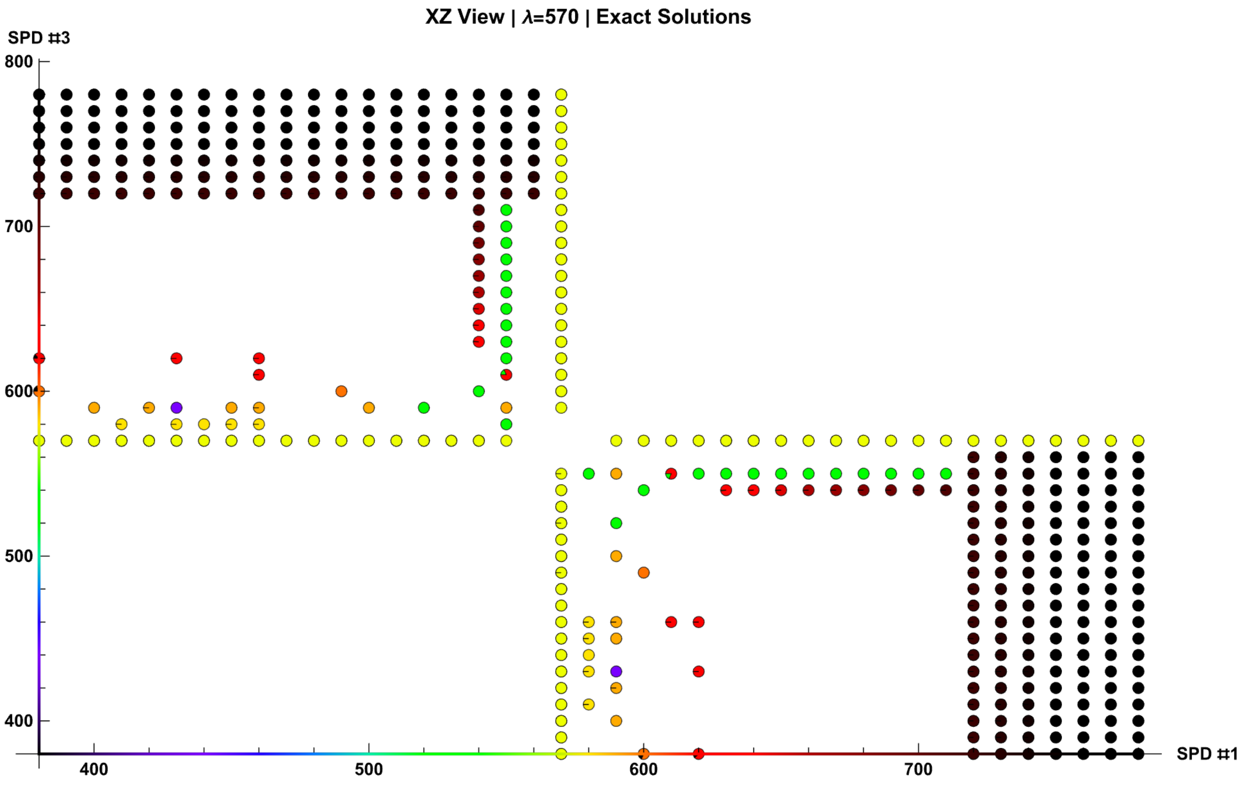
Out[∘]=



Out[ ]=


YZ View | λ=570 | Exact Solutions
SPD #3
800
700
600
500
400
400
500
600
700
SPD #2

Out[ ]=


XZ View | λ=570 | Exact Solutions
SPD #3
800
700
600
500
400
400
500
600
700
SPD #1

Approach #2

```
(*Get the angle in degrees between the two vectors.*)
```

$$\text{degreeBetween[u\_, v\_] := ArcCos}\left[\frac{\text{u.v}}{\text{Norm[u] * Norm[v]}}\right] * \frac{180}{\text{Pi}};$$

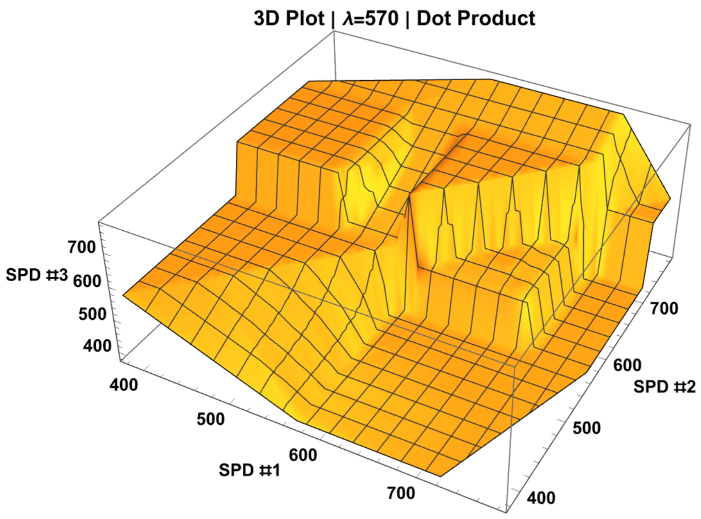
```
(*Set a threshold.*)
degreeMax = 3;

Print["Trials: ", Length[monoLights]];
u := targetTristimulusValues;

metamers = ParallelTable[
    (*Gets the tristimulus values for every light in monoLights.*)
    proposedTristimulusValues = Map[mappings, monoLights⟦i⟧];

    (*Convert so that columns become the tristimulus values.*)
    A = Transpose[proposedTristimulusValues];

    x⃗ = Table[Symbol["k" <> ToString@i], {i, 1, n}];

    v = A.x⃗;

    (*Get degrees between the two vectors.*)
    degree = degreeBetween[u, v];
    solution = FindInstance[degree ≤ degreeMax && AllTrue[x⃗, NonNegative], x⃗, WorkingPrec

    If[
        Length[solution] > 0,
        AssociationThread[monoLights⟦i⟧, Values[solution⟦1⟧]],
    ],
    {i, 1, Length[monoLights]}
];

(*Only select where metamers are found.*)
metamers = Select[metamers, AssociationQ[#] && Length[Keys[#]] == n &];
Print[StringForm["Pair of `` Monochromatic SPDs Metameric to λ=``: ``", n, p₀, Length[meta
```

```
Trials: 12 341

Pair of 3 Monochromatic SPDs Metameric to λ=570: 5815
```

In[ ]:=

```
fullList = Join[metamers, Map[Reverse[#] &, metamers]]; (*Accounting for removed duplicat

ListPlot3D[
 fullList,
 PlotLabel→StringForm["3D Plot | λ=`` | Dot Product", p₀],
 AxesLabel→{"SPD #1", "SPD #2", "SPD #3"},
 LabelStyle→Directive[Bold, Black],
 AxesOrigin → {380, 380}
]
```

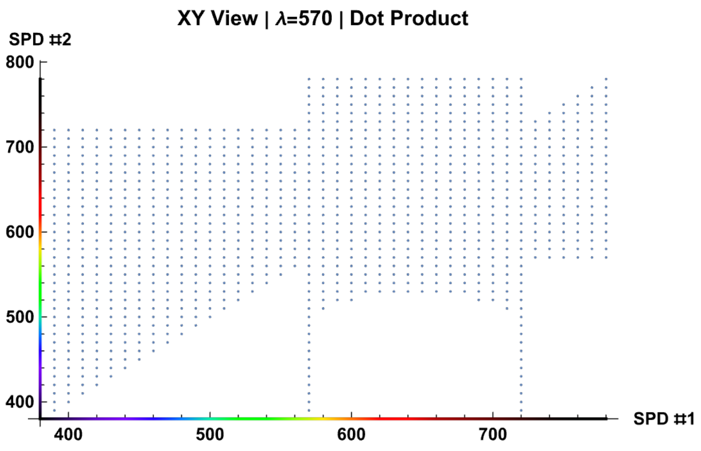
Out[ ]=



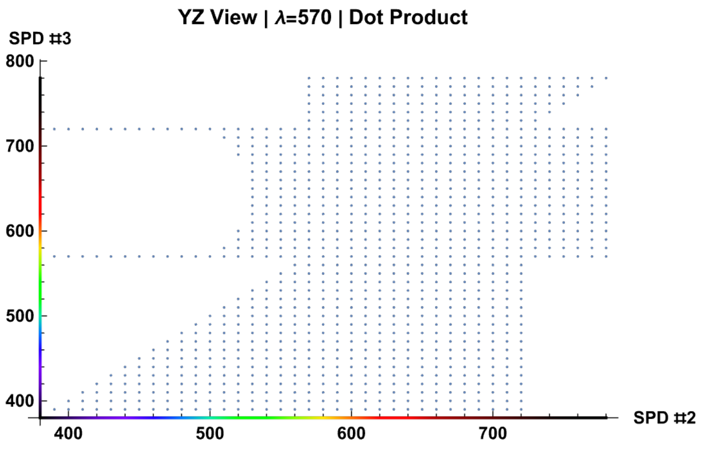
```
subsets = Table[
    AssociationThread[
        #,
        Lookup[fullList[[i]], #]] & /@ Subsets[Keys[fullList[[i]]], {2}],
    {i, Length[fullList]}
];

components = Transpose[subsets];

XY = components[[1]];
colorsXY = Map[ColorData["VisibleSpectrum"], Keys[XY], {2}]; (*List of two-tuple colors b
markersXY = PieChart[XY[[#]], ChartStyle→colorsXY[[#]], ImageSize→7] & /@ Range[Length[XY]];

XZ = components[[2]];
colorsXZ = Map[ColorData["VisibleSpectrum"], Keys[XZ], {2}]; (*List of two-tuple colors b
markersXZ = PieChart[XZ[[#]], ChartStyle→colorsXZ[[#]], ImageSize→7] & /@ Range[Length[XZ]];

YZ = components[[3]];
colorsYZ = Map[ColorData["VisibleSpectrum"], Keys[YZ], {2}]; (*List of two-tuple colors b
markersYZ = PieChart[YZ[[#]], ChartStyle→colorsYZ[[#]], ImageSize→7] & /@ Range[Length[YZ]];
```

```
ListPlot[
 {#} & /@ Keys[XY],
 PlotLabel→StringForm["XY View | λ=`` | Dot Product", p₀],
 AxesLabel→{"SPD #1", "SPD #2"},
 LabelStyle→Directive[Bold, Black],
 Epilog → Style[{
    AbsoluteThickness@1.5,
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{380, x}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ],
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{x, 380}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ]
    }, Antialiasing → False
   ],
 AxesOrigin → {380, 380}
]

ListPlot[
 {#} & /@ Keys[YZ],
 PlotLabel→StringForm["YZ View | λ=`` | Dot Product", p₀],
 AxesLabel→{"SPD #2", "SPD #3"},
 LabelStyle→Directive[Bold, Black],
 Epilog → Style[{
    AbsoluteThickness@1.5,
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{380, x}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ],
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{x, 380}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ]
    }, Antialiasing → False
   ],
 AxesOrigin → {380, 380}
]

ListPlot[
 {#} & /@ Keys[XZ],
 PlotLabel→StringForm["XZ View | λ=`` | Dot Product", p₀],
 AxesLabel→{"SPD #1", "SPD #3"},
 LabelStyle→Directive[Bold, Black],
 Epilog → Style[{
    AbsoluteThickness@1.5,
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{380, x}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ],
    Line[#, VertexColors → #2] & @@ Transpose@Table[
       {{x, 380}, ColorData["VisibleSpectrum"][x]}, {x,
        Subdivide[380, 780, 20]}
       ]
    }, Antialiasing → False
   ],
  AxesOrigin → {380, 380}
 ]
```

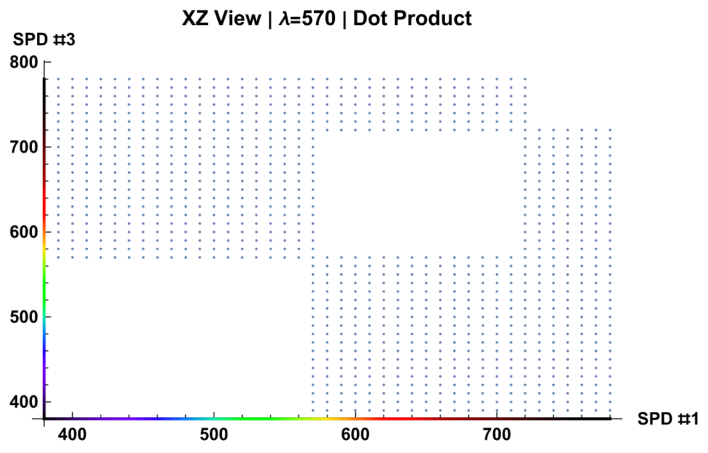
Out[∘]=



Out[∘]=



Out[ ]=



Bibliography

1. Fairchild, M. D. (2005). Colorimetry. In Color appearance models (2 nd ed). J. Wiley.
2. Schanda, J. (2007). CIE Colorimetry. In International Commission on Illumination (Ed.), Colorimetry : Understanding the CIE system. CIE / Commission internationale de l' eclairage ; Wiley – Interscience.
3. Schanda, J. (2007). CIE Colorimetry. In International Commission on Illumination (Ed.), Colorimetry : Understanding the CIE system. CIE / Commission internationale de l' eclairage ; Wiley – Interscience.
4. Wyman C., Sloan P., Shirley P., Simple Analytic Approximations to the CIE XYZ Color Matching Functions. Journal of Computer Graphics Techniques, vol. 2, no. 2, 1 – 11, 2013.