

Rebecca May
Paige Smith
Homework 2 - Chroma Subsampling
Due: 3/27/25

Observations/Comparisons

4:4:4

The original images for the assignment.

4:2:2

The RGB colors on this image seem very similar to 4:4:4's coloring. The range of colors is not noticeably decreased even though they have been, due to the subsampling. Overall, the RGB has maintained its quality well here. In the grayscale image of the Cr layer, there are obvious differences from the original, aside from the fact that it is grayscale. The objects within the photo are noticeably more fuzzy, especially the pills photo. The only time a person can see any difference is when zooming in very closely to the same part of the image for the 4:4:4 and 4:2:2, and this shows subtle discrepancies in coloring of the image content's outlines (for example, the flower petals may look more smooth in 4:4:4 sampling and more rigid in 4:2:2). The grayscale images for the Cr layer compared to the 4:4:4 image or RGB of the 4:2:2 show the image contents as more dull and less in-depth objects. A person can still tell what the image contents are, but some of the pill photo's objects, for example, blend in a little more if they are beside another object of similar color.

4:2:0

This subsampling had more obvious differences for the RGB images than in the 4:2:2 subsampling. When zoomed in on the pills and flower, compared to the 4:4:4 and 4:2:2, one can see there's more blockiness in the 4:2:0 than either of the other two when an object border is looked at (for example, the edges of pill casings or the end of the flower petals). The RGB images look almost the same when not zoomed in for the 4:2:2 and 4:4:4 RGBs as well. For the grayscale images, when compared to the 4:2:2, there are slight changes where one can see the block outlines are slightly larger than the other one, most likely due to a decrease in precision. The grayscale images overall look very similar to the 4:2:2 grayscale.

Conclusions

From experimenting with the different subsampling types, we noticed that every pixel has its own unique color and brightness compared to the original image. The subsampling can be very

useful for transferring high-quality images without the need for high bandwidth to broadcast them, or space to store the picture data. Some good use cases to use the 4:4:4 sampling would be high-end film production, color grading, or visual effects. Use cases for the 4:2:2 subsampling would also be broadcasting or video recording to maintain good quality while decreasing the file size. As for the 4:2:0 subsampling, it would best be used for streaming, DVDs, or Blu-rays for the efficient compression of the picture. For the Y (Luma) component, the resolution is not touched by the subsampling algorithm, thus making it full resolution for the images even after computation. However, for the CbCr (Chroma) component, the resolution is reduced by computing the subsampling algorithm. Although the chroma channel loses resolution, it is very minor to the naked eye.

Subsampling Output Images

4:4:4 - RGB

Original Image - Tulips



Original Image - Pills



4:2:2 - RGB

Tulips



Pills



4:2:0 - RGB

Tulips

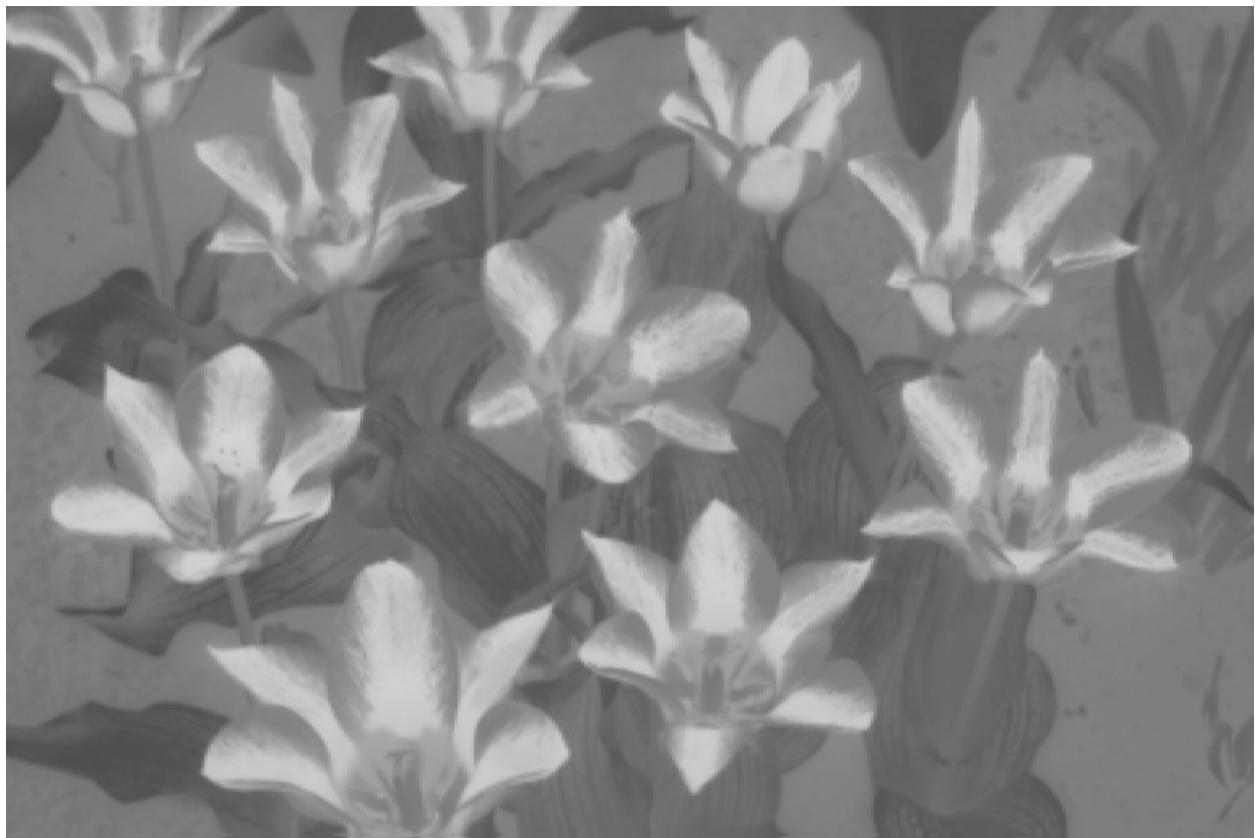


Pills

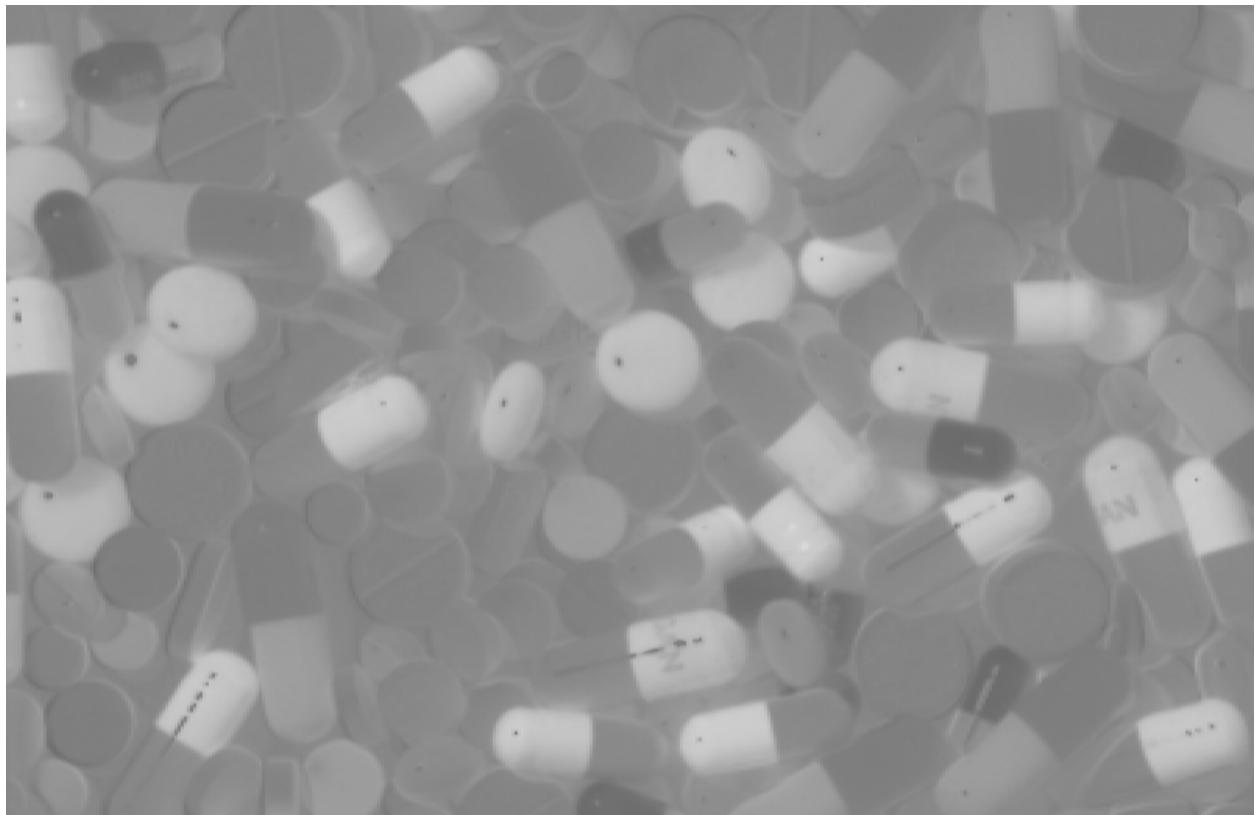


4:2:2 - Grayscale

Tulips

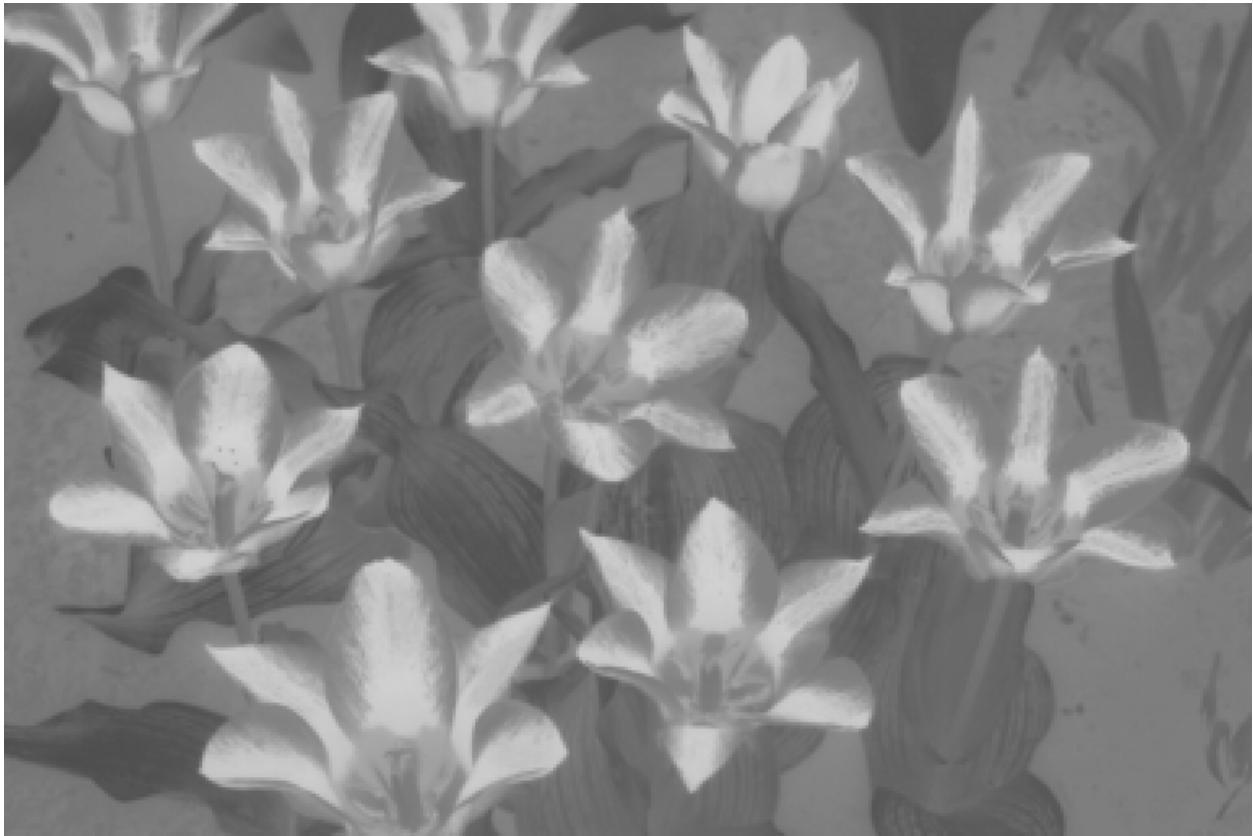


Pills

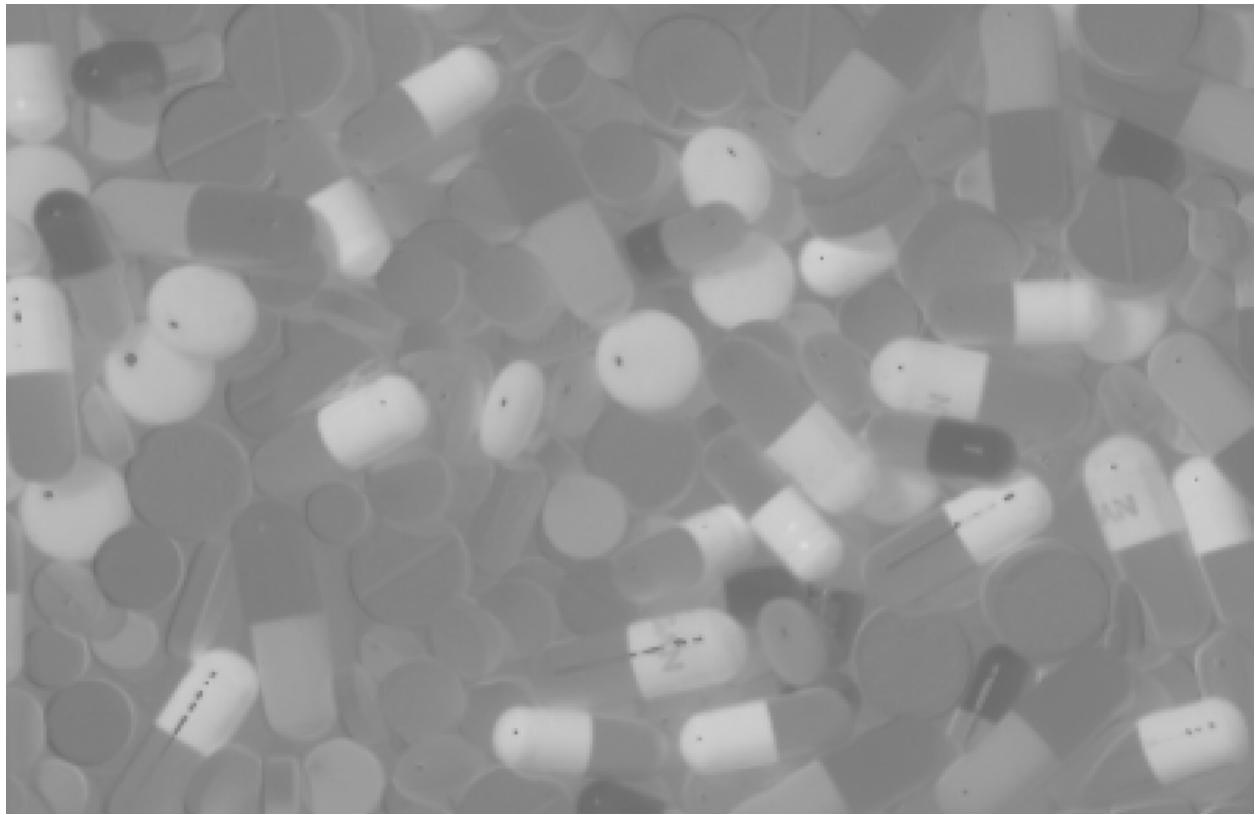


4:2:0 - Grayscale

Tulips



Pills



Block-Wise Output Comparisons

4:2:2

Tulips

0.3961	0.3636	0.3255	0.3204	0.3373	0.3373	0.3412	0.3460
0.3373	0.3382	0.3294	0.3221	0.3255	0.3361	0.3373	0.3550
0.3333	0.3315	0.3255	0.3255	0.3373	0.3412	0.3490	0.3500
0.3176	0.3361	0.3333	0.3334	0.3373	0.3364	0.3373	0.3290
0.3961	0.3569	0.3255	0.3176	0.3373	0.3373	0.3412	0.3451
0.3373	0.3333	0.3294	0.3176	0.3255	0.3373	0.3373	0.3529
0.3333	0.3255	0.3255	0.3255	0.3373	0.3412	0.3490	0.3451
0.3176	0.3333	0.3333	0.3294	0.3373	0.3333	0.3373	0.3255

Pills

0.6510	0.6298	0.6235	0.6215	0.6196	0.6152	0.6000	0.5961
0.6353	0.6348	0.6314	0.6254	0.6275	0.6152	0.6118	0.6044
0.6431	0.6439	0.6314	0.6219	0.6235	0.6106	0.6157	0.6012
0.6471	0.6404	0.6353	0.6286	0.6196	0.6175	0.6078	0.5973
0.6510	0.6275	0.6235	0.6235	0.6196	0.6157	0.6000	0.5961
0.6353	0.6353	0.6314	0.6275	0.6275	0.6157	0.6118	0.6039
0.6431	0.6431	0.6314	0.6235	0.6235	0.6118	0.6157	0.6000

0.6471	0.6392	0.6353	0.6275	0.6196	0.6235	0.6078	0.5961
--------	--------	--------	--------	--------	--------	--------	--------

4:2:0

Tulips

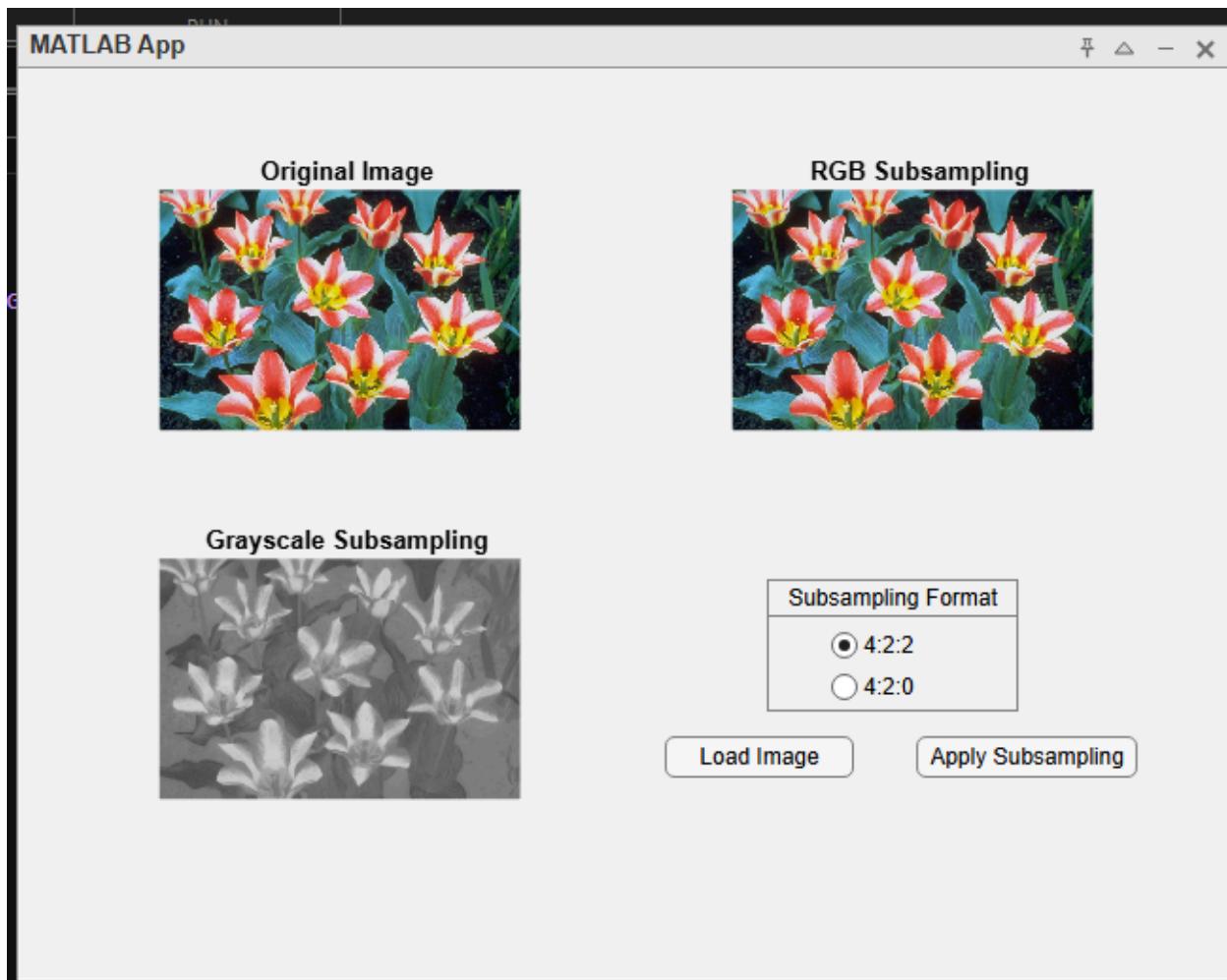
0.3961	0.3636	0.3255	0.3204	0.3373	0.3373	0.3412	0.3460
0.3443	0.3452	0.3318	0.3244	0.3311	0.3417	0.3400	0.3578
0.3333	0.3315	0.3255	0.3255	0.3373	0.3412	0.3490	0.3500
0.3225	0.3410	0.3350	0.3350	0.3415	0.3406	0.3490	0.3407
0.3961	0.3569	0.3255	0.3176	0.3373	0.3373	0.3412	0.3451
0.3373	0.3333	0.3294	0.3176	0.3255	0.3373	0.3373	0.3529
0.3333	0.3255	0.3255	0.3255	0.3373	0.3412	0.3490	0.3451
0.3176	0.3333	0.3333	0.3294	0.3373	0.3333	0.3373	0.3255

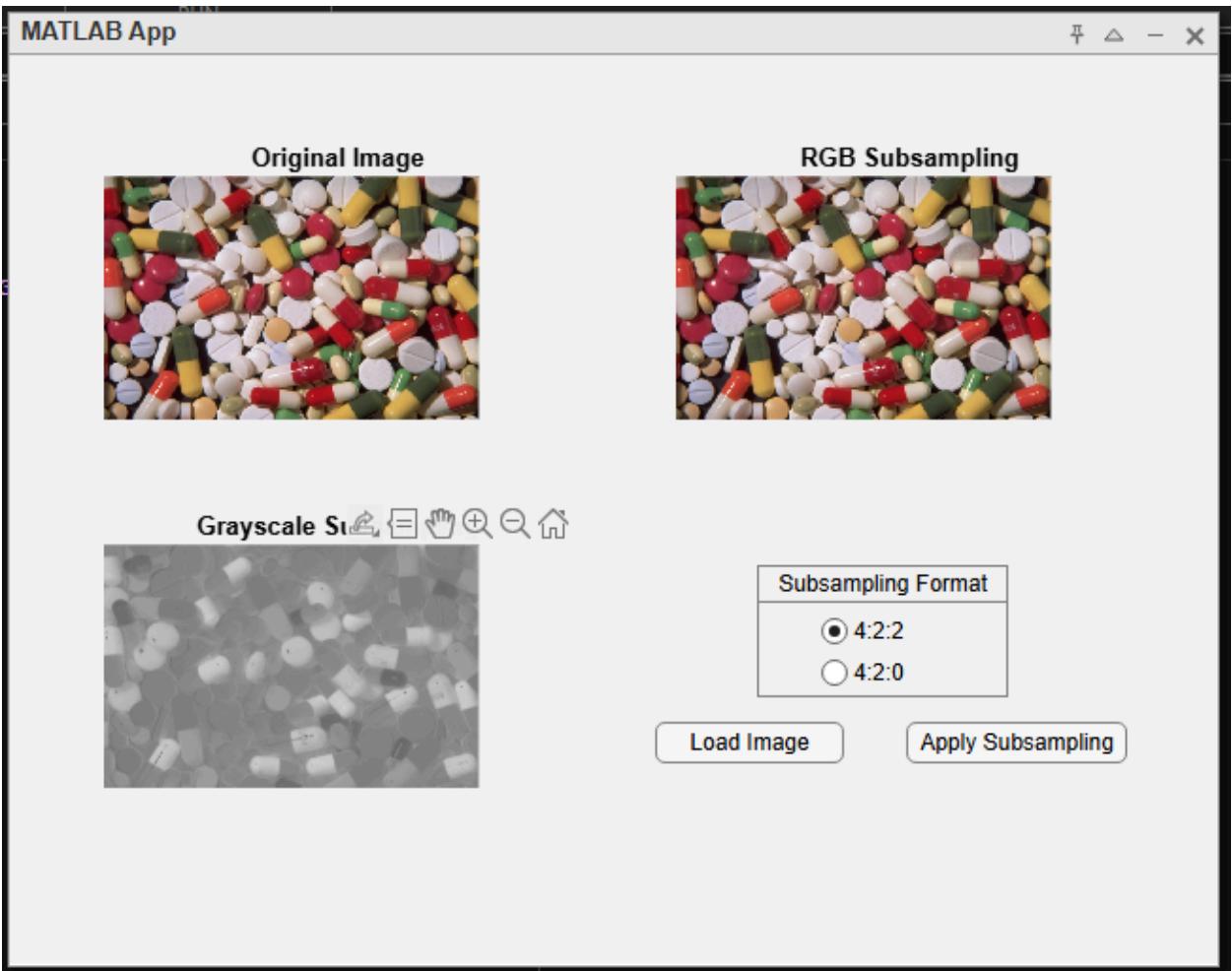
Pills

0.6510	0.6298	0.6235	0.6215	0.6196	0.6152	0.6000	0.5961
0.6381	0.6376	0.6309	0.6249	0.6275	0.6152	0.6113	0.6039
0.6431	0.6439	0.6314	0.6219	0.6235	0.6106	0.6157	0.6012
0.6459	0.6392	0.6341	0.6275	0.6217	0.6196	0.6083	0.5977
0.6510	0.6275	0.6235	0.6235	0.6196	0.6157	0.6000	0.5961
0.6353	0.6353	0.6314	0.6275	0.6275	0.6157	0.6118	0.6039
0.6431	0.6431	0.6314	0.6235	0.6235	0.6118	0.6157	0.6000
0.6471	0.6392	0.6353	0.6275	0.6196	0.6235	0.6078	0.5961

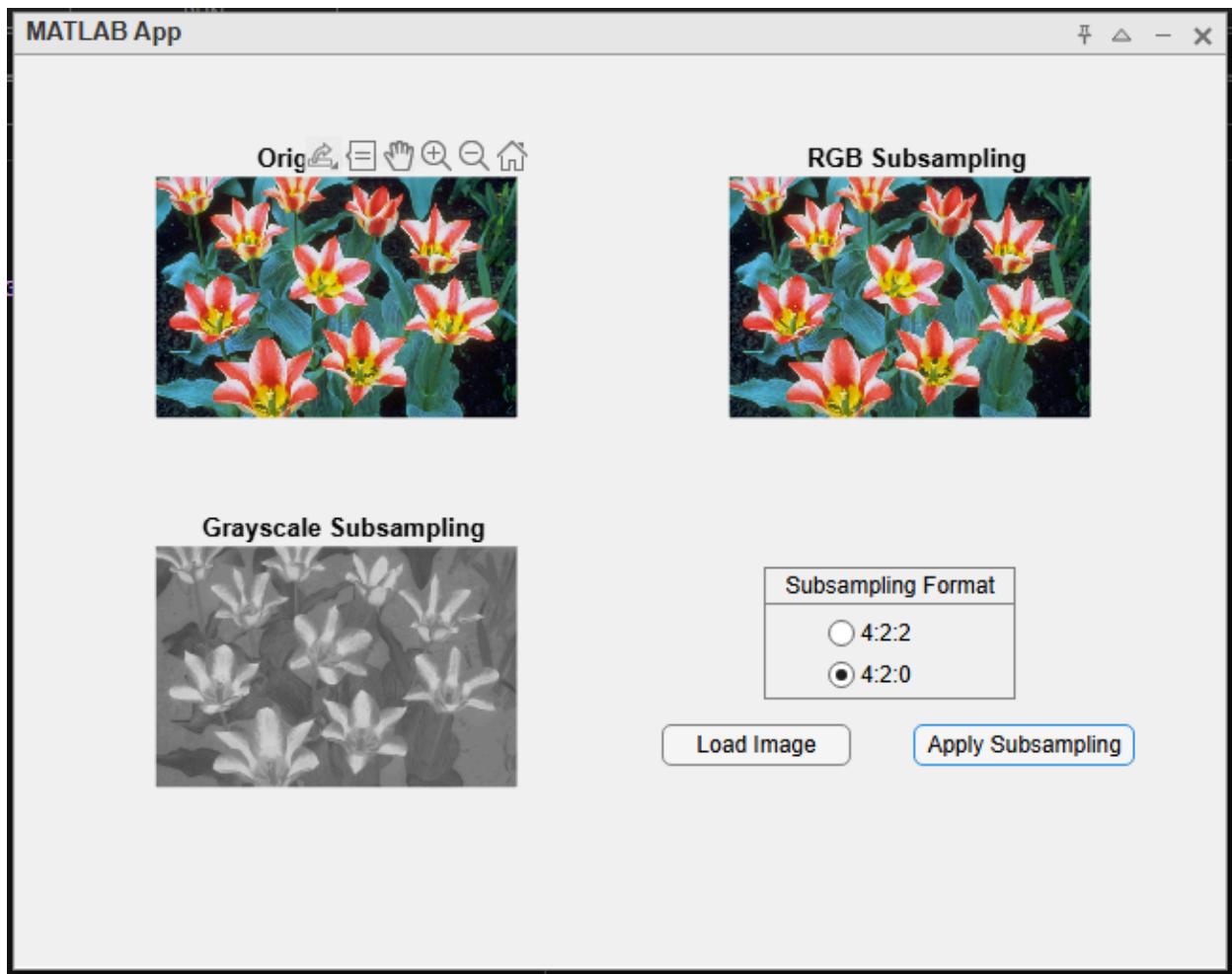
GUI Screenshots

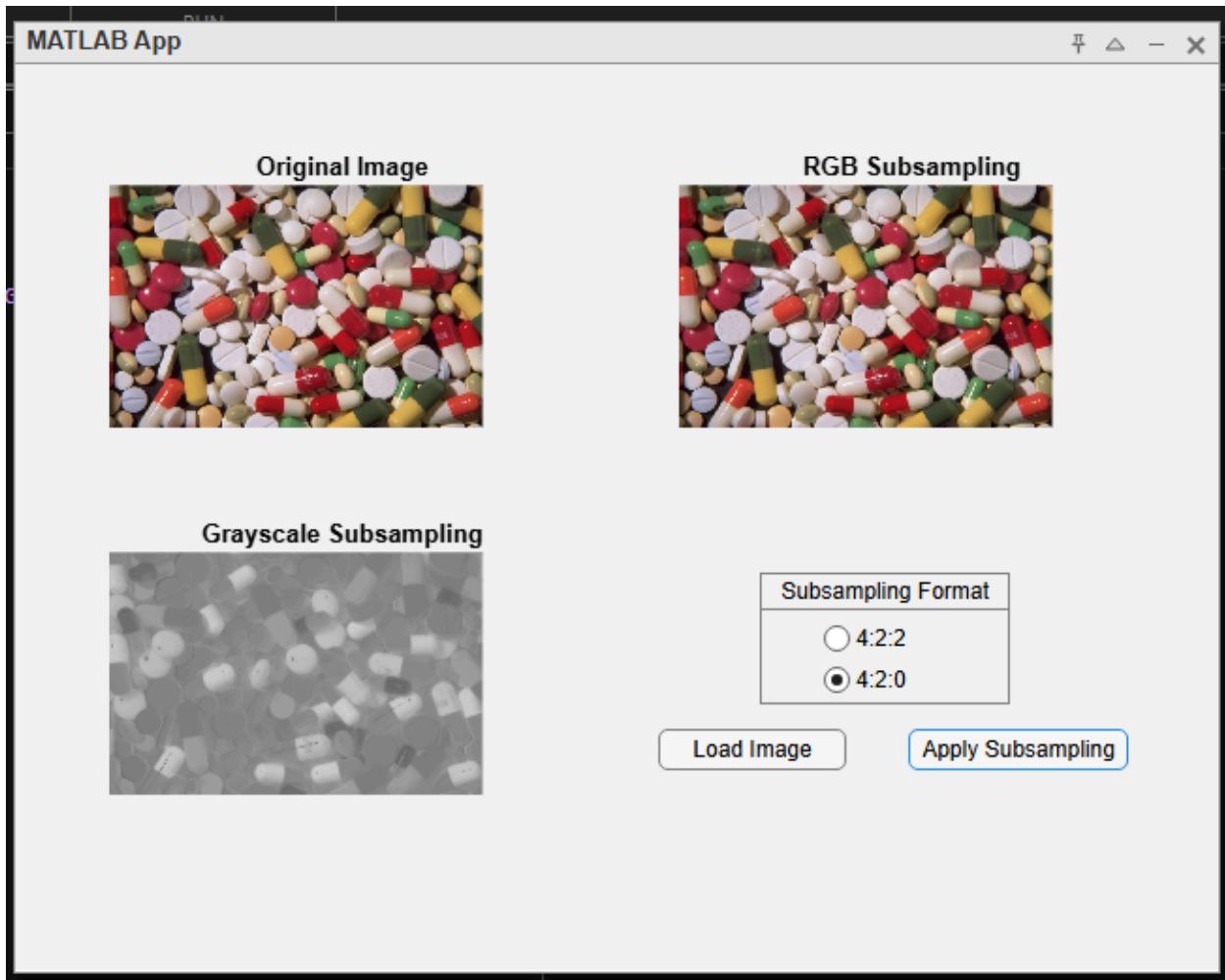
4:2:2





4:2:0





MatLab Source Code

MatLab App Code:

```

classdef subsampling < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                      matlab.ui.Figure
        SubsamplingFormatButtonGroup   matlab.ui.container.ButtonGroup
        Button_2                       matlab.ui.control.RadioButton
        Button                          matlab.ui.control.RadioButton
        ApplySubsamplingButton         matlab.ui.control.Button
        LoadImageButton                matlab.ui.control.Button
        ImageGray                      matlab.ui.control.UIAxes
        ImageRGB                       matlab.ui.control.UIAxes
        ImageOriginal                  matlab.ui.control.UIAxes
    end

```

```

% Callbacks that handle component events
methods (Access = private)
    % Button pushed function: ApplySubsamplingButton
    function apply(app, event)
        % See which subsampling option is selected
        config = false;
        if (app.Button.Value == true)
            % subsampling 4:2:2 chosen
            % call function and apply
            config = true;
        end

        % Call the function that handles subsampling
        [RGB, Gray] = subsample(app.ImageOriginal.UserData, config);

        % display the RGB image on app
        % rgb = imread(RGB);
        axis(app.ImageRGB, 'off');
        set(app.ImageRGB, 'UserData', RGB);
        imshow(RGB, 'Parent', app.ImageRGB);

        % display the grayscale image on app
        % gray = imread(Gray);
        axis(app.ImageGray, 'off');
        set(app.ImageGray, 'UserData', Gray);
        imshow(Gray, 'Parent', app.ImageGray);
    end
    % Button pushed function: LoadImageButton
    function loadImage(app, event)
        [f, p] = uigetfile({'*.jpg; *.png; *.gif', 'All Image Files'});
        if (ischar(p))
            fname = [p f];
            im = imread(fname);
            axis(app.ImageOriginal, 'off');
            set(app.ImageOriginal, 'UserData', im);
            imshow(im, 'Parent', app.ImageOriginal);
        end
    end
end
% Component initialization
methods (Access = private)
    % Create UIFigure and components
    function createComponents(app)
        % Create UIFigure and hide until all components are created
        app.UIFigure = uifigure('Visible', 'off');
        app.UIFigure.Position = [100 100 640 480];
        app.UIFigure.Name = 'MATLAB App';
        % Create ImageOriginal
        app.ImageOriginal = uiaxes(app.UIFigure);

```

```

title(app.ImageOriginal, 'Original Image')
app.ImageOriginal.PlotBoxAspectRatio = [2.07079646017699 1 1];
app.ImageOriginal.XTick = [];
app.ImageOriginal.YTick = [];
app.ImageOriginal.Position = [23 263 304 172];
% Create ImageRGB
app.ImageRGB = uiaxes(app.UIFigure);
title(app.ImageRGB, 'RGB Subsampling')
app.ImageRGB.PlotBoxAspectRatio = [2.07079646017699 1 1];
app.ImageRGB.XTick = [];
app.ImageRGB.YTick = [];
app.ImageRGB.Position = [326 263 304 172];
% Create ImageGray
app.ImageGray = uiaxes(app.UIFigure);
title(app.ImageGray, 'Grayscale Subsampling')
app.ImageGray.PlotBoxAspectRatio = [2.07079646017699 1 1];
app.ImageGray.XTick = [];
app.ImageGray.YTick = [];
app.ImageGray.Position = [23 69 304 172];
% Create LoadImageButton
app.LoadImageButton = uibutton(app.UIFigure, 'push');
app.LoadImageButton.ButtonPushedFcn = createCallbackFcn(app,
@loadImage, true);
app.LoadImageButton.Position = [343 107 100 22];
app.LoadImageButton.Text = 'Load Image';
% Create ApplySubsamplingButton
app.ApplySubsamplingButton = uibutton(app.UIFigure, 'push');
app.ApplySubsamplingButton.ButtonPushedFcn = createCallbackFcn(app,
@apply, true);
app.ApplySubsamplingButton.Position = [476 107 117 22];
app.ApplySubsamplingButton.Text = 'Apply Subsampling';
% Create SubsamplingFormatButtonGroup
app.SubsamplingFormatButtonGroup = uibuttongroup(app.UIFigure);
app.SubsamplingFormatButtonGroup.TitlePosition = 'centertop';
app.SubsamplingFormatButtonGroup.Title = 'Subsampling Format';
app.SubsamplingFormatButtonGroup.Position = [397 142 133 70];
% Create Button
app.Button = uiradiobutton(app.SubsamplingFormatButtonGroup);
app.Button.Text = '4:2:2';
app.Button.Position = [34 23 66 22];
app.Button.Value = true;
% Create Button_2
app.Button_2 = uiradiobutton(app.SubsamplingFormatButtonGroup);
app.Button_2.Text = '4:2:0';
app.Button_2.Position = [34 1 66 22];
% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end

```

```

% App creation and deletion
methods (Access = public)
    % Construct app
    function app = subsampling
        % Create UIFigure and components
        createComponents(app)
        % Register the app with App Designer
        registerApp(app, app.UIFigure)
        if nargout == 0
            clear app
        end
    end
    % Code that executes before app deletion
    function delete(app)
        % Delete UIFigure when app is deleted
        delete(app.UIFigure)
    end
end

```

MatLab .m Code:

```

% Rebecca May, Paige Smith
% Homework 2
% Due: 3/27/25
% Create a MatLab function that converts an image's RGB values to
% YCbCr and applies subsampling to it.
% This function will be called in the MatLab subsampling application
function [outputImgRGB, outputImgGray] = subsample(inputImg, config)
    % Grab the image and put its values in a matrix
    OrigImage = im2double(inputImg);
    [row, col, layer] = size(OrigImage);
    % YCbCr Matrix (given in assignment)
    ConvertYCbCr = [0.299 0.587 0.114; -0.16874 -0.33126 0.5; 0.5 -0.41869
    -0.08131];
    % Column YCbCr aid matrix (given in assignment)
    YCbCrAid = [0; 0.5; 0.5];

    % Convert to YCbCr
    ConvertedImage = zeros(row, col, layer);
    % Iterate through the rows and columns and convert to YCbCr
    for x = 1:row
        for y = 1:col
            % Separate the RGB layers and perform conversion
            OrgRGB = [OrigImage(x, y, 1); OrigImage(x, y, 2); OrigImage(x,
            y, 3)];
            YCbCr = (ConvertYCbCr * OrgRGB) + YCbCrAid;
            % Put layers back in one matrix to apply subsampling later
        end
    end

```

```

        ConvertedImage(x, y, 1) = YCbCr(1, 1);
        ConvertedImage(x, y, 2) = YCbCr(2, 1);
        ConvertedImage(x, y, 3) = YCbCr(3, 1);
    end
end
% Perform subsampling
Subsampled = ConvertedImage;
% Determine which subsample was chosen
if (config)
    % 4:2:2 is chosen when true
    % Loop over the matrix in 2x4 blocks
    for r = 1:2:row
        for c = 1:4:col
            % Safe indexing to stay within bounds
            r2 = min(r+1, row);      % 2nd row
            c2 = min(c+1, col);      % 2nd column
            c3 = min(c+2, col);      % 3rd column
            c4 = min(c+3, col);      % 4th column

            % ===== Cb Layer =====
            % Copy cell with with into corresponding cells in 2x4
            Subsampled(r, c2, 2) = ConvertedImage(r, c, 2);
            Subsampled(r, c4, 2) = ConvertedImage(r, c3, 2);
            Subsampled(r2, c2, 2) = ConvertedImage(r2, c, 2);
            Subsampled(r2, c4, 2) = ConvertedImage(r2, c3, 2);

            % ===== Cr Layer =====
            % Copy cell with with into corresponding cells in 2x4
            Subsampled(r, c2, 3) = ConvertedImage(r, c, 3);
            Subsampled(r, c4, 3) = ConvertedImage(r, c3, 3);
            Subsampled(r2, c2, 3) = ConvertedImage(r2, c, 3);
            Subsampled(r2, c4, 3) = ConvertedImage(r2, c3, 3);
        end
    end
else
    %4:2:0 is chosen when false
    % Loop over the matrix in 2x4 blocks
    for r = 1:2:row
        for c = 1:4:col
            % Compute safe indices within bounds
            r2 = min(r+1, row);      % 2nd row
            c2 = min(c+1, col);      % 2nd column
            c3 = min(c+2, col);      % 3rd column
            c4 = min(c+3, col);      % 4th column

            % ===== Cb Layer =====
            % Copy cell with with into corresponding cells in 2x4
            Subsampled(r, c2, 2) = ConvertedImage(r, c, 2);
            Subsampled(r2, c, 2) = ConvertedImage(r, c, 2);

```

```

Subsampled(r2, c2, 2) = ConvertedImage(r, c, 2);
Subsampled(r, c4, 2) = ConvertedImage(r, c3, 2);
Subsampled(r2, c3, 2) = ConvertedImage(r, c3, 2);
Subsampled(r2, c4, 2) = ConvertedImage(r, c3, 2);

% ===== Cr Layer =====
% Copy cell with with into corresponding cells in 2x4
Subsampled(r, c2, 3) = ConvertedImage(r, c, 3);
Subsampled(r2, c, 3) = ConvertedImage(r, c, 3);
Subsampled(r2, c2, 3) = ConvertedImage(r, c, 3);
Subsampled(r, c4, 3) = ConvertedImage(r, c3, 3);
Subsampled(r2, c3, 3) = ConvertedImage(r, c3, 3);
Subsampled(r2, c4, 3) = ConvertedImage(r, c3, 3);
end
end
% Make a copy of the YCbCr converted image for output2
% Separate the layers from Cr subsampled image
CrImage(:, :) = Subsampled(:, :, 3);
outputImgGray = CrImage;
% Save the grayscale output image
imwrite(outputImgGray, "outputImgGray.png");

% RGB Matrix (given in assignment)
ConvertRGB = [1 0 1.402; 1 -0.34414 -0.71414; 1 1.77200 0];
outputImgRGB = zeros(row, col, layer);
% Iterate through subsampled matrix and convert back to RGB values
for x = 1:row
    for y = 1:col
        % Subtract 0.5 from Cb and Cr rows only
        RGBAid = [Subsampled(x, y, 1); Subsampled(x, y, 2)-0.5;
        Subsampled(x, y, 3)-0.5];

        % Convert the subsampled matrix back to RGB Form
        RGB = (ConvertRGB)*(RGBAid);
        RGB = max(0, min(255, RGB));
        % Add R and G and B layers back
        outputImgRGB(x, y, 1) = RGB(1, 1);
        outputImgRGB(x, y, 2) = RGB(2, 1);
        outputImgRGB(x, y, 3) = RGB(3, 1);
    end
end
% Save RGB output image
imwrite(outputImgRGB, "outputImgRGB.png");
% Display for .m running testing purposes
disp(outputImgRGB(1:4, 1:8, 2))
disp(OrigImage(1:4, 1:8, 2))
end

```