数字图象处理

课程作业二

2019010485　自 91　刘祖炎*

2021 年 10 月 9 日

1　实现原理与代码分析

1.1 算法原理

本次作业的算法原理较为简单。具体而言实现了两个函数，分别为根据当前状态计算当前图片值的函数 *calc_current_image* 和计算图片滤波后结果的函数 *filter_image*。具体代码如下：

```
function result = filter_image(app, image)
    filter = fspecial('average', round(27/(app.ASlider.Value)/(app.ASlider.Value)+1));
    result = imfilter(image, filter, 'same', 'conv');
end
```

filter_image 函数的输入为图片 *image*，输出为经过滤波后的图片 *result*。利用函数 *fspecial* 实现均值滤波器，并利用 *imfilter* 函数对图像进行滤波。其中，滤波器半径大小的计算公式为：

$$R = [\frac{27}{A(A+1)}]$$

这样，当光圈值 A 增大时，滤波器的卷积核半径逐渐减小，使得图像更加清晰。

```
function calc_current_image(app)
    curr_ISO = app.ISOSlider.Value;
    curr_A = app.ASlider.Value;
    curr_S = app.SSlider.Value;
    curr_EV = -log2((100 * curr_A * curr_A) / (curr_ISO * curr_S));
    delta_EV = curr_EV - app.init_EV;
    delta_pix = delta_EV / 9;
    if app.FocusButtonGroup.SelectedObject.Text == 'Foreground'
        curr_foreground = app.foreground + delta_pix;
        curr_background = app.filter_image(app.background) + delta_pix;
    else
        curr_foreground = app.filter_image(app.foreground) + delta_pix;
        curr_background = app.background + delta_pix;
```

*liuzuyan19@mails.tsinghua.edu.cn

```
    end
    app.curr_image = curr_background .* (1 - app.mask) + curr_foreground .* app.mask;
    app.Image.ImageSource = app.curr_image;
end
```

calc_current_image 函数用于根据当前各个相机参数值更新当前图像值。

首先从各个调节组件中获得当前 ISO、A、S 值，并根据上述值与公式 (1) 计算当前 EV 值，此后，根据图像初始 EV 值与公式 (2) 计算当前像素值与原图的差值。此处，由于图像 RGB 值的取值范围为 $[0,1]$，故公式前系数应为 1。按作业要求，取强度系数 $k=9$。根据焦点不同，取不同的图像进行模糊处理，若焦点为前景图片 Θ_1，则利用均值滤波器 $\mathcal{F}$ 模糊处理背景图片 Θ_2，并利用前景遮罩 M 得到当前图片，如公式 (3)；若焦点为背景图片 Θ_2，则利用均值滤波器 $\mathcal{F}$ 模糊处理前景图片 Θ_1，并利用前景遮罩 M 得到当前图片，如公式 (4)。完成图片处理后，得到 *curr_image*，更新至图像类中的 *ImageSource* 变量即可刷新当前图片。

$$EV = -log_2 \frac{100A^2}{ISO \times S} \tag{1}$$

$$\Delta pix = \frac{\Delta EV}{k} \tag{2}$$

$$\Theta = M \cdot \Theta_1 + \sim M \cdot \mathcal{F}(\Theta_2) \tag{3}$$

$$\Theta = M \cdot \mathcal{F}(\Theta_1) + \sim M \cdot \Theta_2 \tag{4}$$

1.2 *UI* 界面

软件整体界面如图1所示。

UI 界面利用 Matlab 提供的 *AppDesigner* 实现。

左上角下拉框用于选取展示的图片，可在 *Image*1(作业提供的图片) 与 *Image*2(自己准备的图片) 之间切换。

Start 按钮用于启动软件，在右侧显示图片。

Focus 单选按钮组用于切换焦点位置至前景图与背景图处。

ISO、*A*、*S* 滑条用于改变当前相机的对应参数，在各自的右上角会显示当前参数，其中，感光度 *ISO* 的调节范围为 $[100, 6400]$，光圈 A 的调节范围为 $[1.8, 32]$，快门 S 的调节范围为 $[1/2048, 2]$。

最下方会显示当前图像的 EV 值，右侧显示当前图片。

1.3 界面代码分析

由于 Matlab 的 *AppDesigner* 已经实现了 *app* 类与大多数函数，故此处只分析各个 UI 界面的回调函数。

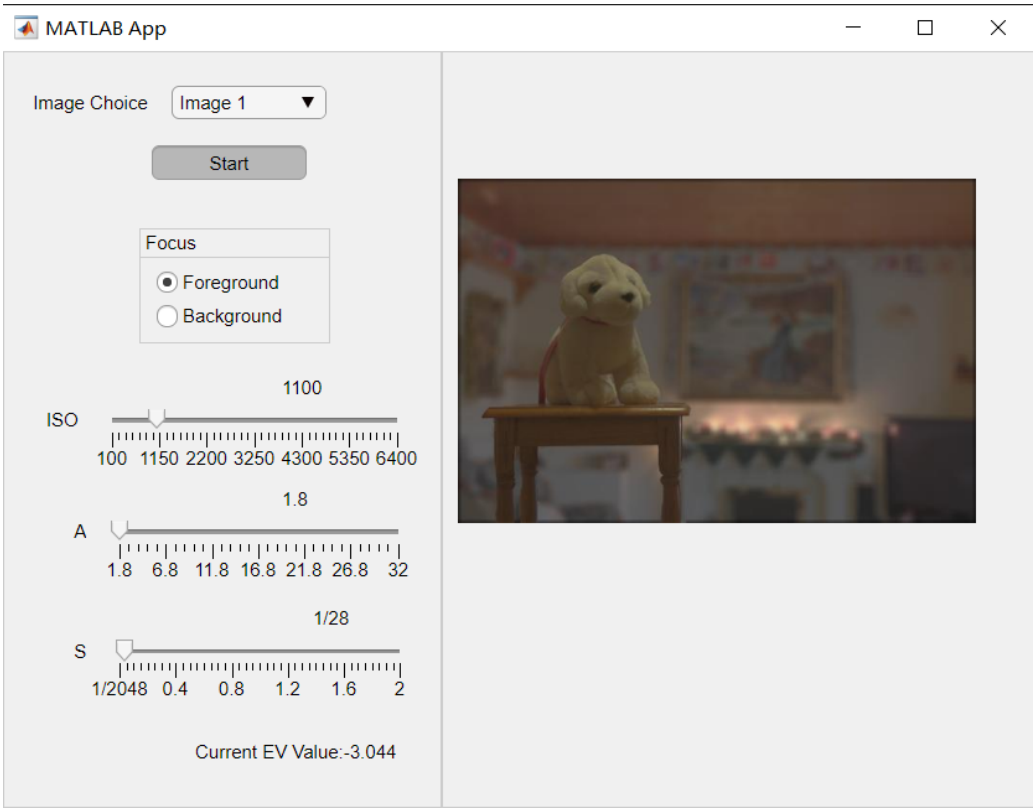

图 1: UI 界面

1.3.1 成员变量

首先，需要声明类 *private* 类型成员变量供各函数使用：

```
properties (Access = private)
    foreground
    background
    mask
    curr_image
    init_EV = -log2((100 * 1.8 * 1.8) / (1100 * (1 / 28)));
end
```

foreground 与 *background* 变量分别对应前景与背景图的原图

mask 为前景遮罩

$curr_image$ 为当前图像

$init_EV$ 为当前图片的原始 *EV* 值

1.3.2 初始化函数

初始化函数在完成组件初始化后执行。代码如下：

```
function startupFcn(app)
    app.ISOValueLabel.Text = num2str(app.ISOSlider.Value);
    app.AValueLabel.Text = num2str(app.ASlider.Value);
    app.SValueLabel.Text = num2str(rats(app.SSlider.Value));
```

```
    [app.foreground, ~, app.mask] = imread('lounge-hdr-foreground.png');
    app.background = imread('lounge-hdr-background.png');
    app.foreground = im2double(app.foreground);
    app.background = im2double(app.background);
    app.mask = im2double(app.mask);
    app.EVValueLabel.Text = ['Current EV Value:', num2str(roundn(app.init_EV, -3))];
    app.Image.ImageSource = zeros(size(app.foreground));
end
```

初始化时，更新各滑条右上角标签值为默认值，读取图片 1 并将其转换为 *double* 类型，更新 *EV* 标签为当前 *EV* 值，初始化图片为全黑背景图。

此处需要注意以下几点：

- 快门值要利用 *rats* 函数将浮点数转换为分数形式，与习惯相符。
- 读取前景 *png* 图片时，不应使用一般的读取方式，而应当利用 Matlab 自带的功能将 png 的 RGB 图像与其 *alpha* 通道分开，分别储存在 *foreground* 与 *mask* 变量中，这样既可以节省计算前景遮罩的计算量，也可以避免在后续图片处理的滤波时出现黑色边框，使图像更加自然。
- 显示 *EV* 值时，利用 *roundn* 函数进行四舍五入，-3 表示保留三位小数。

1.3.3 启动按钮回调函数

启动按钮回调函数在点击启动按钮后执行。代码如下：

```
function StartButtonValueChanged(app, event)
    value = app.StartButton.Value;
    if value == 1
        calc_current_image(app);
    end
    if value == 0
        app.Image.ImageSource = zeros(size(app.foreground));
    end
end
```

其逻辑较为简单，若按钮开启，则更新当前图片；若按钮关闭，则将背景设置为全黑。

1.3.4 焦点单选按钮回调函数

焦点单选按钮回调函数在改变焦点位置后执行，代码如下：

```
function FocusButtonGroupSelectionChanged(app, event)
    if app.StartButton.Value == 1
        calc_current_image(app);
    end
end
```

由于在 *calc_current_image* 函数中已实现根据焦点更新图片的功能，故此处在开启按钮启动的基础上执行该函数即可。

1.3.5 图像选择框回调函数

图像选择框回调函数在改变图像选择框内容后执行，代码如下：

```
function ImageChoiceDropDownValueChanged(app, event)
    value = app.ImageChoiceDropDown.Value;
    if value == 'Image 1'
        app.ISOSlider.Value = 1100;
        app.ASlider.Value = 1.8;
        app.SSlider.Value = 1/28;
        [app.foreground, ~, app.mask] = imread('lounge-hdr-foreground.png');
        app.background = imread('lounge-hdr-background.png');
        app.init_EV = -log2((100 * 1.8 * 1.8) / (1100 * (1 / 28)));
    elseif value == 'Image 2'
        app.ISOSlider.Value = 100;
        app.ASlider.Value = 1.8;
        app.SSlider.Value = 1/100;
        [app.foreground, ~, app.mask] = imread('forground_1.png');
        app.background = imread('background_1.png');
        app.init_EV = -log2((100 * 1.8 * 1.8) / (100 * (1 / 100)));
    end
    app.ISOValueLabel.Text = num2str(app.ISOSlider.Value);
    app.AValueLabel.Text = num2str(app.ASlider.Value);
    app.SValueLabel.Text = num2str(rats(app.SSlider.Value));
    app.foreground = im2double(app.foreground);
    app.background = im2double(app.background);
    app.mask = im2double(app.mask);
    app.EVValueLabel.Text = ['Current EV Value:', num2str(roundn(app.init_EV, -3))];
    if app.StartButton.Value == 1
        calc_current_image(app);
    end
end
```

根据当前选择值为图片 1 或图片 2，设置不同的 ISO、A、S、EV 值，并读取对应的图片。此后，按照与初始化函数相同的方式对图像进行处理，更新 EV 的标签值。若当前软件已启动，则更新图像。

1.3.6 ISO 滑条回调函数

ISO 滑条回调函数在调节滑条位置时执行，代码如下：

```
function ISOSliderValueChanging(app, event)
    changingValue = event.Value;
    app.ISOValueLabel.Text = num2str(round(changingValue));
    if app.StartButton.Value == 1
        calc_current_image(app);
    end
end
```

若当前软件已启动，则更新图像。根据对感光度的习惯设置，将当前值四舍五入取整后更新为 ISO 标签值。

1.3.7 A 滑条回调函数

A 滑条回调函数在调节滑条位置时执行，代码如下：

```
function ASliderValueChanging(app, event)
    changingValue = event.Value;
    app.AValueLabel.Text = num2str(roundn(changingValue, -1));
    if app.StartButton.Value == 1
        calc_current_image(app);
    end
end
```

若当前软件已启动，则更新图像。根据对光圈值的习惯设置，将当前值保留一位小数舍入后更新为 A 标签值。

1.3.8 S 滑条回调函数

S 滑条回调函数在调节滑条位置时执行，代码如下：

```
function SSliderValueChanging(app, event)
    changingValue = event.Value;
    if changingValue > 0.1
        app.SValueLabel.Text = num2str(roundn(changingValue, -1));
    else
        intValue = num2str(round(1 / changingValue));
        app.SValueLabel.Text = ['1/', intValue];
    end

    if app.StartButton.Value == 1
        calc_current_image(app);
    end
end
```

若当前软件已启动，则更新图像。根据对快门速度的习惯设置，若当前快门时间大于 0.1 秒，则直接保留一位小数舍入后更新为 S 标签值；若当前快门时间小于 0.1 秒，则将快门时间从小数换算为 $1/n$ 的形式。

2 实验结果与分析

经过实验验证，该软件能够完美地运行并显示预期结果，具有较强的鲁棒性与可交互性。展示部分运行结果如下图所示。

在未点击启动按钮前，调节各个部件不产生作用，点击启动按钮后，显示对应图像：

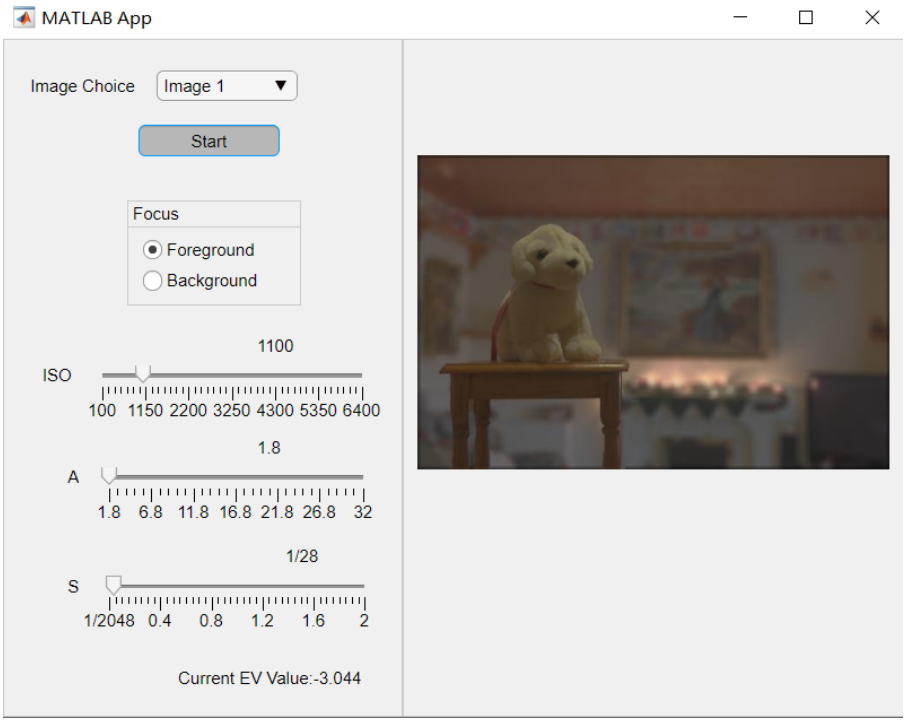

图 2: 启动

改变焦点，明显看到前景变模糊，背景变清晰。

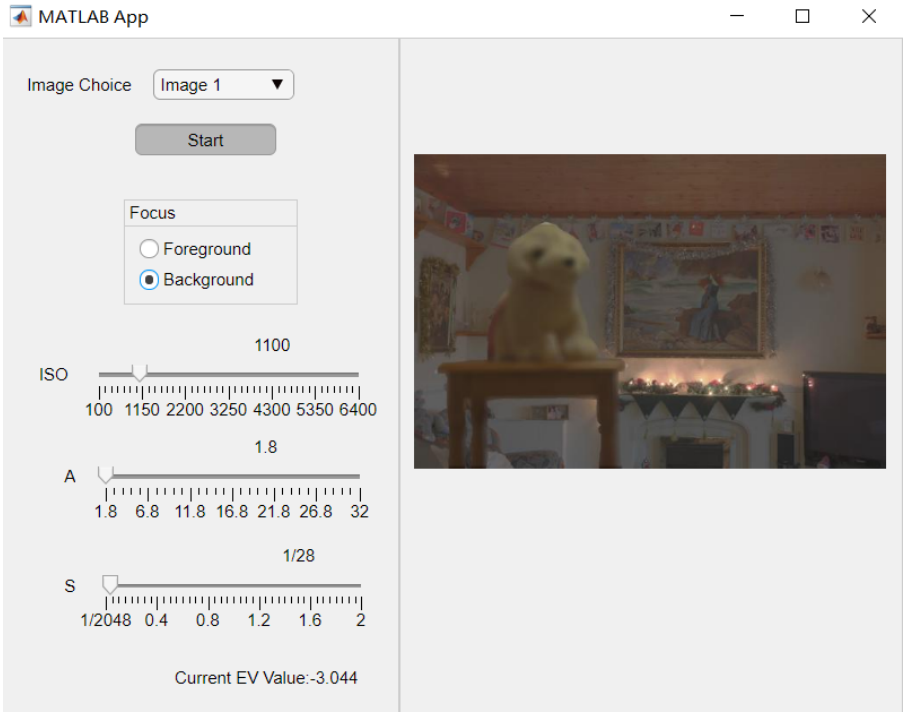

图 3: 调节焦点

提高 ISO，观察到图像变亮，EV 值增大。

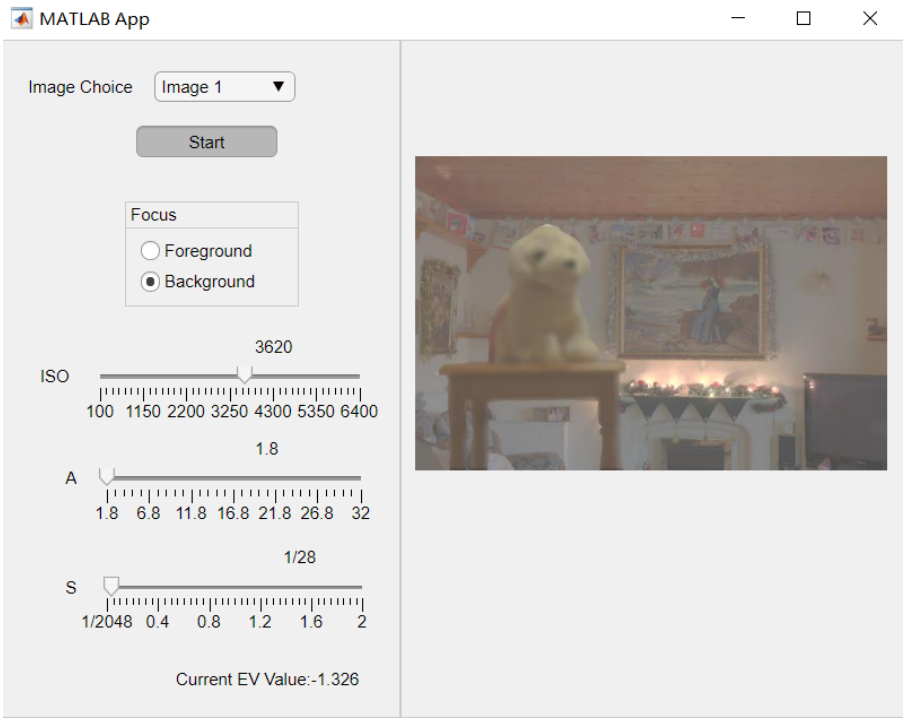

图 4: 改变 ISO

减小光圈，观察到图像变暗，EV 值减小。

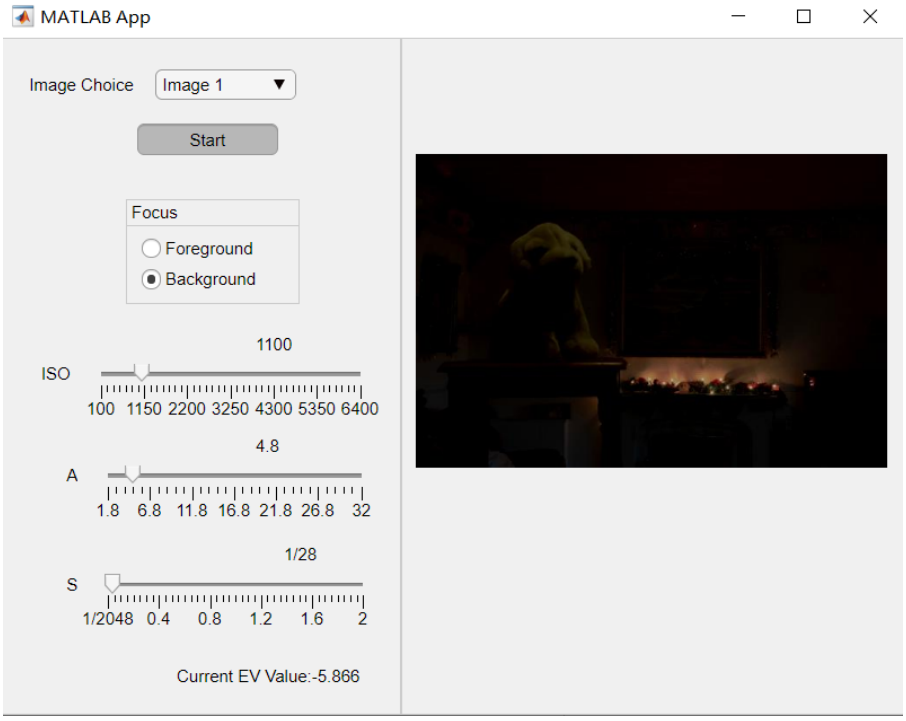

图 5: 改变光圈

增大快门时间，观察到图像变量，EV 值增大。

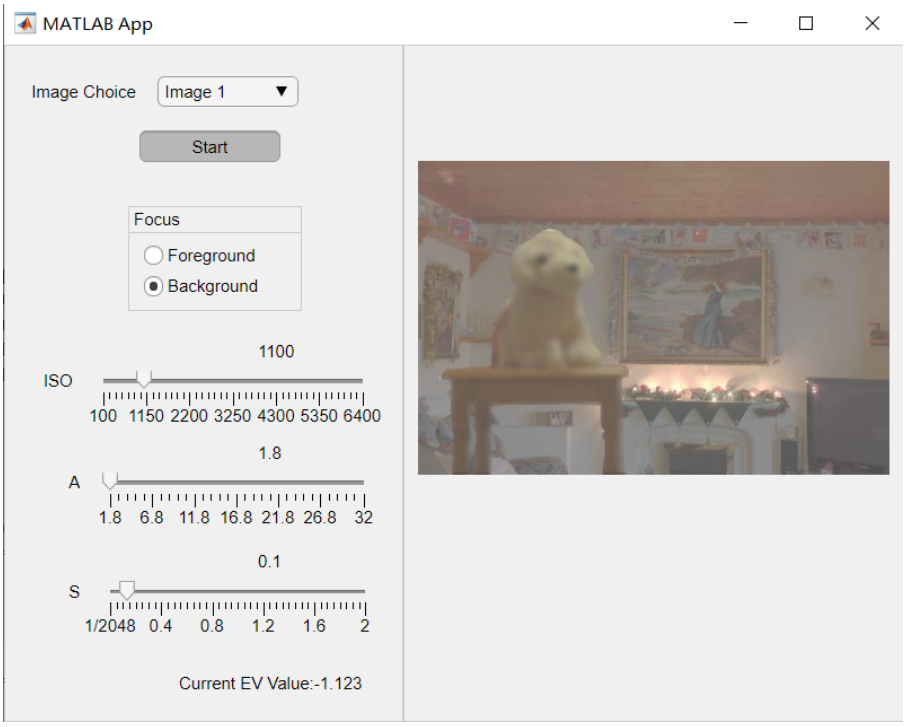

图 6: 增大快门时间

减小光圈后，调节其他参数使图片恢复原始 EV 值，观察到前景变清晰，表示景深变浅。

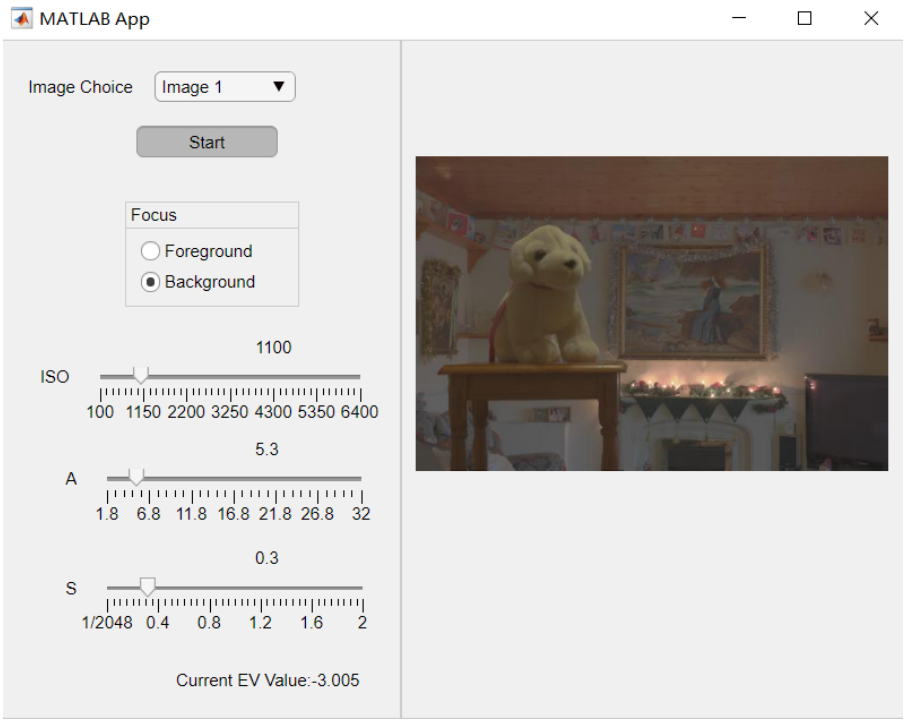

图 7: 景深变化

切换图片后，观察到图片切换为图片 2。

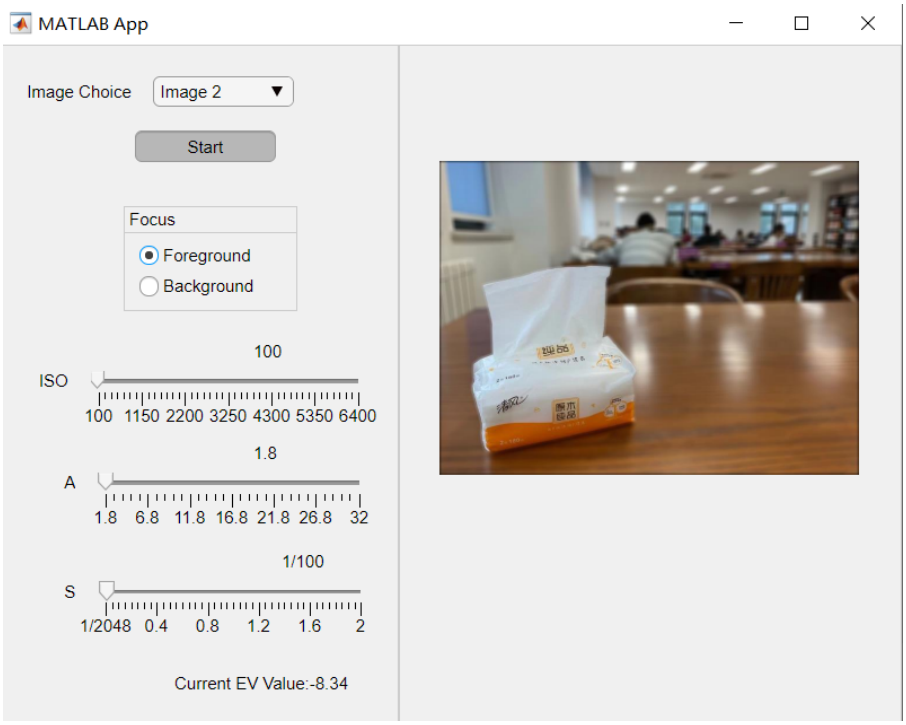

图 8: 图片切换

改变焦点，明显看到前景变模糊，背景变清晰。

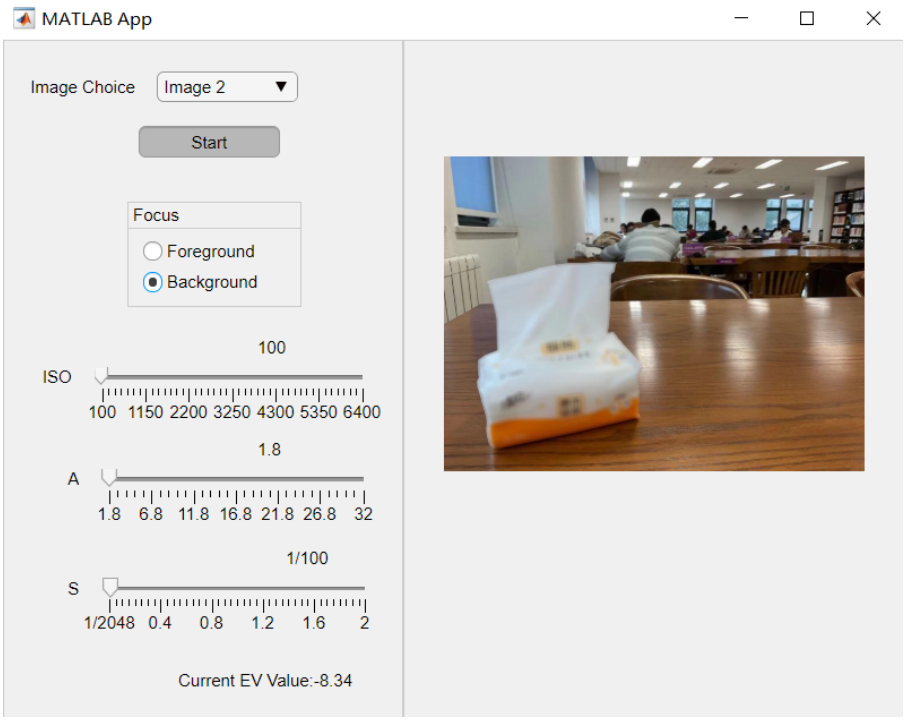

图 9: 调节焦点

3 遇到的困难和解决办法

本次作业代码逻辑较为简单，在编程时并未遇到困难。主要解决了以下几个问题：

- **Matlab 软件 UI 界面的使用。**笔者发现 *AppDesigner* 比 *GUIDE* 更为便捷，易用性与界面美观

程度均更高。在掌握了 *AppDesigner* 的用法后，笔者很快便完成了各个函数的编写。

- *png* **图片的读取。**笔者在开始时发现经过滤波处理后的图片有黑框，这是因为具有透明度设置的 *png* 在读入 Matlab 后会自动将透明部分转为黑色部分。笔者通过尝试发现了可以利用不同的赋值方式将 RGB 通道与 *Alpha* 通道分开，使得在处理 RGB 图像时不会处理到边缘处的黑框，完美地解决了该问题。
- **软件测试。**笔者花费较多时间对软件进行测试，确保软件能够在各种条件与操作下正常运行。

4 收获

- Matlab 实现图形界面的方法。
- Matlab 实现图像滤波的方法。
- 相机参数设置以及摄影原理的应用。

5 可能的改进方向

- 除手动模式 M 外，可基于图像 EV 值不变的原理实现光圈优先、快门优先、自动模式等调节。