数字图像处理
第一次小作业报告

自96 曲世远 2019011455

1.

本次作业要求实现图片转场（变换）的效果，我除了基本要求的3+3种以外，还额外实现了一种灰度变换的效果（溶解），一种集合组合变换（旋转+缩放）。

2.代码及算法解释

```
%% read and resize and renew max_Y
for i = 1:len
    image_dir = [image_files(i).folder, '\', image_files(i).name];
    image_read = imread(image_dir);% read image
    image_read = im2double(image_read);%uint to double
    image_read = imresize(image_read, [600, nan]);% resize
    image_collect{i} = image_read;
    a = size(image_read);
    max_Y = max(max_Y, a(2));%renew max_Y
    %imshow(image_read);
end

%% fill the background with black
for i = 1:len
    a = size(image_collect{i});
    if a(2) == max_Y
        continue
    else
        back_bg = zeros(600, max_Y, 3); % black ground
        l = (max_Y / 2) - (a(2) / 2);
        back_bg(:, l : max_Y - l - 1, :) = image_collect{i}(:, :, :);% why
this index contains both right and left! make my code not beautiful!
        image_collect{i} = back_bg;
        %imshow(back_bg);
    end
end
```

首先，对图片文件读入后重新改变大小，并填充背景。在这一部分，只需要运用matlab中的基本函数与基本的矩阵元素操作，就可以实现。

```
%% 棋盘格淡入淡出 （灰度变换类）
image_idx = 2;
block_size = 120;
block_index = zeros(600, max_Y);
for i = 1: 600 % generate block_index
    for j = 1: max_Y
        if (mod(idivide(int32(i), int32(block_size), 'floor'), 2) == ...
                mod(idivide(int32(j), int32(block_size), 'floor'), 2))
            block_index(i, j) = 1;
        else
        end
```

```
    end
end
%imshow(block_index)
image_gen = image_collect{image_idx - 1};
% image_gen = image_gen + image_collect{image_idx} .* block_index;
% imshow(image_gen .* ~block_index)
%stage one
for factor = 0: 0.04: 1
    image_gen = image_gen .* ~block_index + ...
        image_collect{image_idx} .* block_index * factor + ...
        image_collect{image_idx - 1} .* block_index * (1 - factor);
    imwrite(image_gen, [new_main_dir, num2str(save_idx,'%05d'),file_type]);
    save_idx = save_idx + 1;
%     imshow(image_gen);
%     pause(0.001);
end
%stage two
for factor = 0: 0.04: 1
    image_gen = image_gen .* block_index + ...
        image_collect{image_idx} .* ~block_index * factor + ...
        image_collect{image_idx - 1} .* ~block_index * (1 - factor);
    imwrite(image_gen, [new_main_dir, num2str(save_idx,'%05d'),file_type]);
    save_idx = save_idx + 1;
%     imshow(image_gen);
%     pause(0.001);
end
```

本段代码实现的是棋盘格淡入淡出的变换。由于棋盘格的位置不好通过矩阵索引确定，所以我构造了一个与图片等大的矩阵，通过布尔值标定棋盘格的位置，这样就可以通过矩阵运算快速的实现棋盘格淡入淡出的效果。

```
%% 按方向淡入淡出 （灰度变换类）
image_idx = 3;
block_index = 0 : 1 / 400 : 1;
block_index = [zeros(1, max_Y), block_index, ones(1, max_Y)];
block_size = 401;
for i = 1 : 20 :max_Y + block_size
    block_temp = block_index(max_Y + block_size - i + 2 : max_Y + block_size
- i + 1 + max_Y);
    block_temp = repmat(block_temp, 600, 1);
    image_gen = image_collect{image_idx - 1} .* block_temp + ...
        image_collect{image_idx} .* (ones(600, max_Y) - block_temp);
%     imshow(image_gen);
%     pause(0.001);
    imwrite(image_gen, [new_main_dir, num2str(save_idx,'%05d'),file_type]);
    save_idx = save_idx + 1;
end
```

本段代码实现的是按方向淡入淡出的变换。由于淡入淡出的效果想要更好的表现，需要生成一个灰度值渐变的区域块，但仍需要考虑越界等情况。由此，我构造了一个$2 \times max_Y + blocksize$长的行向量，通过将这个行向量平移截取，得到$blocksize$的相对移动，再将截取的部分扩展为矩阵，就可以得到一个淡入淡出的变换矩阵，于是就可以通过矩阵运算漂亮的实现淡入淡出的效果。

```
%% 溶解 （灰度变换类）
image_idx = 5;
image_gen = image_collect{image_idx - 1};
```

```
block_index = randperm(600 * max_Y);
block_size = 1;
for scale = 0.02 : 0.02 : 1
    while block_size < scale * 600 * max_Y
        [a, b] = ind2sub([600, max_Y], block_index(block_size));
        image_gen(a, b, :) = ...
            image_collect{image_idx}(a, b, :);
        block_size = block_size + 1;
    end
%     imshow(image_gen);
%     pause(0.001);
    imwrite(image_gen, [new_main_dir, num2str(save_idx,'%05d'),file_type]);
    save_idx = save_idx + 1;
end
```

本段代码是我额外实现的溶解效果。这一效果实现的要点在于生成一个随机乱序且不重复的矩阵索引。于是我使用了randperm函数，再将数值索引转换为下标索引，从而实现逐像素的图片替换，也就是溶解的效果。

```
%% 缩放动画 （几何变化类）
image_idx = 6;
image_gen = image_collect{image_idx - 1};
for scale = 0.02: 0.02: 1
    image_temp = imresize(image_collect{image_idx}, scale);
    image_size = size(image_temp);
    l_x = idivide(int32(600 - image_size(1)), 2);
    l_y = idivide(int32(max_Y - image_size(2)), 2);
    if (l_x | l_y) == 0
        l_x = 1;l_y = 1;
    end
    image_gen(int32(l_x) : int32(l_x + image_size(1) - 1), int32(l_y) :
int32(l_y + image_size(2) - 1), :) = image_temp;
%     imshow(image_gen);
%     pause(0.001);
    imwrite(image_gen, [new_main_dir, num2str(save_idx,'%05d'),file_type]);
    save_idx = save_idx + 1;
end
```

本段代码实现的是缩放动画的效果。缩放动画比较简单，只需要将图片进行imresize就可以，主要在于将缩放后的图片放置在背景图片合适的位置，有可能会出现非整数的情况，因此需要适当取整及确定范围。

```
%% 单页翻转动画 （几何变化类）
image_idx = 7;
image_idx = image_idx - 1;
for i = [max_Y : -20 : 20, 40 : 20 : max_Y] % stage 1
    image_gen = zeros(600, max_Y, 3);
    if i == 20
        image_idx = image_idx + 1;
    end
    image_temp = imresize(image_collect{image_idx}, [600, i]);
    l = idivide(int32(max_Y - i), int32(2));
    image_gen(:, l + 1 : l + i, :) = image_temp;
%     imshow(image_gen);
%     pause(0.001);
    imwrite(image_gen, [new_main_dir, num2str(save_idx,'%05d'),file_type]);
```

```
    save_idx = save_idx + 1;
end
```

本段代码实现的是单页翻转动画效果。实际上只需要横向缩窄目标图片即可，所以操作方式与上一种动画类似。

```
%% 平移动画 （几何变化类）
image_idx = 8;
block_index = [zeros(1, max_Y), ones(1, max_Y)];
for pos = 1 : 20 : max_Y + 1
    block_temp = block_index(pos : pos + max_Y - 1);
    block_temp = repmat(block_temp, 600, 1);
    image_gen = image_collect{image_idx} .* block_temp + ...
        image_collect{image_idx - 1} .* ~block_temp;
%     imshow(image_gen);
%     pause(0.001);
    imwrite(image_gen, [new_main_dir, num2str(save_idx,'%05d'),file_type]);
    save_idx = save_idx + 1;
end
```

本段代码实现了照片的平移变换。这一动画的实现原理与方向淡入淡出类似，可以通过构造一个长行向量截取并扩展为矩阵，再通过矩阵运算即可实现该功能。

```
%% 旋转+缩放动画 （几何变化类）
image_idx = 10;
image_idx = image_idx - 1;
for scale = [1 : -0.02 : 0.02, 0.04 : 0.02 : 1] %stage 1
    image_gen = zeros(600, max_Y, 3);
    if scale == 0.02
        image_idx = image_idx + 1;
    end
    image_temp = imrotate(imresize(image_collect{image_idx}, scale), ...
        mod(int32(scale * 360), int32(360)), "bilinear", "loose");
    image_size = size(image_temp);
    if image_size(1) > 600
        l_x = idivide(int32(image_size(1) - 600), 2);
        image_size(1) = 600;
        image_temp = image_temp(l_x : l_x + 599, :, :);
    end
    if image_size(2) > max_Y
        l_y = idivide(int32(image_size(2) - max_Y), 2);
        image_size(2) = max_Y;
        image_temp = image_temp( : ,l_y : l_y + max_Y - 1, :);
    end
    l_x = idivide(int32(600 - image_size(1)), 2);
    l_y = idivide(int32(max_Y - image_size(2)), 2);
    if l_x == 0
        l_x = 1;
    end
    if l_y == 0
        l_y = 1;
    end
    image_gen(int32(l_x) : int32(l_x + image_size(1) - 1), ...
        int32(l_y) : int32(l_y + image_size(2) - 1), :) = image_temp;
%     imshow(image_gen);
%     pause(0.001);
```

```
    imwrite(image_gen, [new_main_dir, num2str(save_idx,'%05d'),file_type]);
    save_idx = save_idx + 1;
end
```

本段代码实现的是旋转+缩放的功能。主要的难点在于旋转后的图像可能会超出原有的图片范围，需要精确的考虑截取的范围。首先需要正确的使用imrotate函数，保留旋转后的完整图片，之后通过读取大小，进行适当裁剪，得到旋转后的图片与背景的叠加，即可实现本功能。

3.收获及感想

本次实验我首次使用了matlab，在助教上课时认真的教导下很快的熟悉了matlab的语法和函数用法，较为顺利地实现了本次作业要求。经过本次作业较为深刻的感想就是矩阵的运算确实可以将一些算法很快速漂亮的实现。比如本次作业中我最为得意的就是淡入淡出的实现方式，我通过构造一个行向量进行滑动截取与扩展，通过简洁的矩阵运算就实现了功能要求。