多媒体技术作业 1

数据科学与计算机学院 向泳郦 16340249

■ 第一题

① 题目描述

写出一个 openning circle 的视频转场效果。

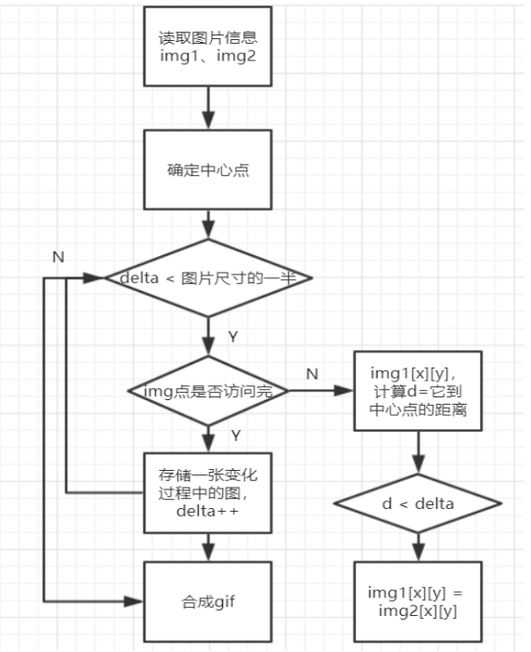
② 算法设计

我们把底图记作 img1,上层图记作 img2。首先我们需要确定 transition 时圆的中心点记作（X, Y)。为了衡量每一时刻有多大的 img1 被变为 img2，我们引入变量 delta（delta 是边界到中心点的距离)。在任意一时刻，如果像素点位于圆圈内，像素信息为 img2 的，否则是 img1 的。

伪代码：

```
FOR delta FROM 0 TO width/2
   FOR EYERY POINT IN imag
       IF (the distance between point and center point) < delta
           POINT of imag = POINT of img2
       ELSE
           POINT of imag = POINT of img1
```

算法框图：



③ 实验环境

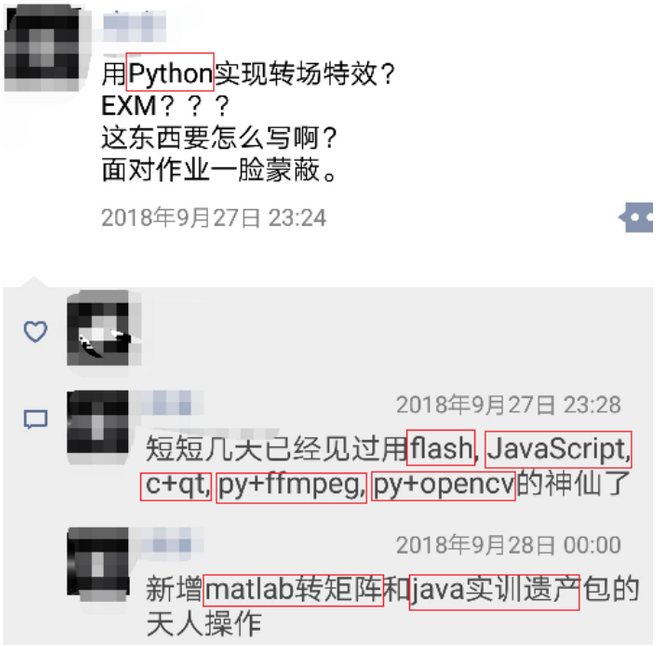
Windows + pycharm + python

④ 代码实现

```
import numpy as np
from PIL import Image
import imageio
# gif 动图生成
def create_gif(image_list, gif_name):
    frames = []
    for image_name in image_list:
        frames.append(imageio.imread(image_name))
    imageio.mimsave(gif_name, frames, 'GIF', duration=0.1)

    return

# 读取 Imag 信息
img1 = Image.open("lena.jpg").convert("RGB")
img2 = Image.open("nobel.jpg").convert("RGB")
image1 = np.array(img1)
height1, width1 = img1.size
image2 = np.array(img2)
height2, width2 = img2.size

#确定中心点
X = int(width2 / 2)
Y = int(height2 / 2)
delta = 0
i = 0


#图片专场像素信息的变化
for delta in range(0, int(height2/2), int(height2/10)):
    for y in range(Y - delta, Y + delta):
        for x in range(X - delta, X + delta):
            if np.power((x - X), 2)+np.power((y - Y), 2) < np.power(delta, 2):
                image2[y][x] = image1[y][x]
    i=i+1
    Image.fromarray(image2).save('circle' + str(i) + '.png')

#gif 动图
image_list = ['circle1.png', 'circle2.png', 'circle3.png',
              'circle4.png', 'circle5.png', 'circle6.png']

gif_name = 'circle.gif'
create_gif(image_list, gif_name)
```

⑤ 实验难点

1. 实验环境的选择：一开始拿到问题真的很无从下手，某同学的朋友圈生动形象地给我们说明了我们对于这道题语言环境选择的迷茫。最后决定采用具有强大封装性的 python 语言。由于 python 语言具有许许多多有用的包，因此通过查询我使用了 PIL+imageio+numpy 成功解决了这道题。PIL 用于读取图片信息并转换为矩阵，numpy 用于进行相应的算术操作，imageio 则让我成功合成了 gif 动图。



2. 此外问题一的主要问题在于写出 transition 时像素变换的式子，如下：

```
#图片专场像素信息的变化
for delta in range(0, int(height2/2), int(height2/10)):
    for y in range(Y - delta, Y + delta):
        for x in range(X - delta, X + delta):
            if np.power((x - X), 2)+np.power((y - Y), 2) < np.power(delta, 2):
                image2[y][x] = image1[y][x]
```

⑥ 实验效果

成果 gif 图放在‘实验结果’文件夹中。

左为 gif6 帧截图

■ 第二题

① 题目描述

For the color LUT problem, try out the median-cut algorithm on a sample image. Explain briefly why it is that this algorithm, carried out on an image of red apples, puts more color gradation in the resulting 24-bit color image where it is needed, among the reds.

② 算法设计

整体思路是：把一整个图像读进去然后就按照课本里的 median cut 把一种颜色从小到大排，然后取中点，把像素点切分为两块，如此一直切到 256 份，每一份根据 LUT 选取一种代表色，得到一个 8bit 的模糊图。

Median-cut:

（1） 找出最小的方块，它包含图像中所有颜色
（2） 沿着方块的长边，从小到大排列它所包含的颜色
（3） 在排序后的中间处将该方块分为两部分
（4） 重复（2）（3）步骤，直至初始方块被划分为 256 个区域
（5） 在每一个区域，将该区域 R、G、B 的平均值作为中心代表颜色
（6） 计算像素点与该区域中心点 RGB 值的欧式距离，给每个像素分配一个代表颜色。
（7） 在指向代表颜色的 LUT 表中用编号代替像素，得到一个 256 行的表。一个 8bit 的模糊图

③ 实验环境

Linux + tck/tk 脚本语言

④ 实现代码

```
package provide mediancut

namespace eval mediancut {
    namespace export reduce
}

proc mediancut::reduce {src dest depth} {

    variable new

    set new(count) 0

    set w [image width $src]
    set h [image height $src]

    set pixList [list]

    for {set y 0} {$y < $h} {incr y} {
```

```
        for {set x 0} {$x < $w} {incr x} {

            lappend pixList [$src get $x $y]
        }
    }

    subdivide $pixList $depth

    apply $src $dest

    return $new(count)
}

proc mediancut::subdivide {pixList depth} {

    variable new

    set num [llength $pixList]

    for {set i 0} {$i < 256} {incr i} {
        set n(r,$i) 0
        set n(g,$i) 0
        set n(b,$i) 0
    }

    foreach pix $pixList {
        foreach {r g b} $pix break
        incr n(r,$r)
        incr n(g,$g)
        incr n(b,$b)
    }

    # Work out which colour has the widest range

    foreach col [list r g b] {

        set l($col) [list]

        for {set i 0} {$i < 256} {incr i} {

            if { $n($col,$i) != 0 } {
                lappend l($col) $i
            }
        }
```

```
        set range($col) [expr {[lindex $l($col) end] - [lindex $l($col) 0]}]
    }

    if { $depth == 0 || ($range(r) == 0 && $range(g) == 0 && $range(b) == 0) } {

        # Average colours

        # puts "Average colour for $num pixels"
        # puts "Range: $range(r) $range(g) $range(b)"

        foreach col [list r g b] {
            set tot 0
            foreach entry $l($col) {
                incr tot [expr {$n($col,$entry) * $entry}]
            }
            set av($col) [expr {$tot / $num}]
        }

        set newpixel [list $av(r) $av(g) $av(b)]
        set fpixel [format "#%02x%02x%02x" $av(r) $av(g) $av(b)]

        # puts "Colour: $newpixel"

        foreach entry $pixList {

            set new($entry) $fpixel
        }
        incr new(count)

    } else {

        # Find out which colour has the maximum range (green, red, blue in order of import
ance)
        set maxrange -1
        foreach col [list g r b] {

            if { $range($col) > $maxrange } {
                set splitcol $col
                set maxrange $range($col)
            }
        }

        # Now work out where to split it
```

```
        set thres [expr {$num / 2}]

        set pn 0
        set tn 0
        set pl [lindex $l($splitcol) 0]

        foreach tl $l($splitcol) {

            incr tn $n($splitcol,$tl)

            if { $tn > $thres } {

                if { $tn - $thres < $thres - $pn } {
                    set cutnum $tl
                } else {
                    set cutnum $pl
                }
                break
            }

            set pn $tn
            set pl $tl
        }

        # puts "Need to split $splitcol at $cutnum"

        # Now split the pixels into the 2 lists

        set hiList [list]
        set loList [list]

        set i [lsearch [list r g b] $splitcol]
        foreach entry $pixList {
            if { [lindex $entry $i] <= $cutnum } {
                lappend loList $entry
            } else {
                lappend hiList $entry
            }
        }

        incr depth -1

        subdivide $loList $depth
        subdivide $hiList $depth
```

```
    }
}

proc mediancut::apply {src dest} {

    variable new

    set w [image width $src]
    set h [image height $src]
    $dest configure -width $w -height $h

    for {set y 0} {$y < $h} {incr y} {

        set row [list]

        for {set x 0} {$x < $w} {incr x} {

            lappend row $new([$src get $x $y])
        }
        $dest put -to 0 $y [list $row]
        update idletasks
    }
}
```

⑤ 实验效果