Homework8 CP

胡成成 2101210578

Question

三选一：

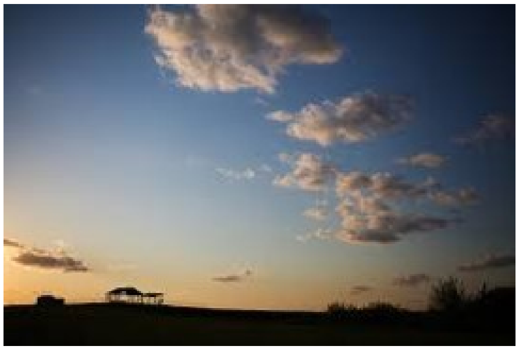
1. 修正图中的暗角；
2. 采用高动态范围技术融合图像；
3. 采用高分辨率技术扩大图像。

Answer

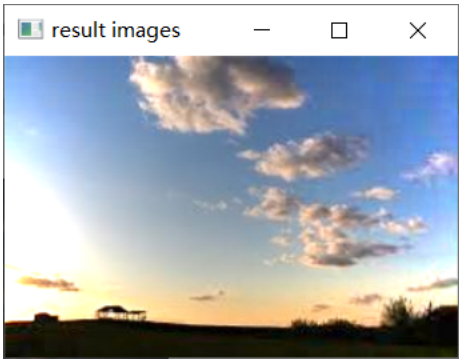
- 选择作业一，修正图片的暗角
- 代码：

```
import cv2
import os
import math
import numpy as np
from scipy.ndimage import gaussian_filter1d


def check_monotonically_increase(parameter_tup):
    """Check if a, b, c could let g(r) monotonically increase in [0,1]"""
    a, b, c = parameter_tup
    if c == 0:
        if a >= 0 and a + 2 * b >= 0 and not(a == 0 and b == 0):
            return True
        return False
    if c < 0:
        if b**2 > 3 * a * c:
            q_plus = (-2 * b + math.sqrt(4 * b**2 - 12 * a * c)) / (6 * c)
            q_minus = (-2 * b - math.sqrt(4 * b**2 - 12 * a * c)) / (6 * c)
            if q_plus <= 0 and q_minus >= 1:
                return True
        return False
    if c > 0:
        if b**2 < 3 * a * c:
            return True
        elif b**2 == 3 * a * c:
            if b >= 0 or 3 * c + b <= 0:
                return True
        else:
            q_plus = (-2 * b + math.sqrt(4 * b**2 - 12 * a * c)) / (6 * c)
            q_minus = (-2 * b - math.sqrt(4 * b**2 - 12 * a * c)) / (6 * c)
            if q_plus <= 0 or q_minus >= 1:
                return True
        return False


def calc_discrete_entropy(cm_x, cm_y, max_distance, parameter_tup, im):
    """
    Calculate the discrete entropy after the brightness of the picture is
    adjusted by the gain function with given parameters    """
    print(parameter_tup)
    a, b, c = parameter_tup
    row, col = im.shape
    histogram = [0 for i in range(256)]
    for i in range(col):
        for j in range(row):
            # calculate the distance from the current pixel to picture's center of mass and the corresponding r value
            distance = math.sqrt((i - cm_x)**2 + (j - cm_y)**2)
            r = distance / max_distance
            # evaluate the gain function and adjust pixel luminance value
            g = 1 + a * r**2 + b * r**4 + c * r**6
            intensity = im[j, i] * g
            # map the luminance value to the corresponding histogram bins
            bin = 255 * math.log(1 + intensity) / math.log(256)
            floor_bin = math.floor(bin)
            ceil_bin = math.ceil(bin)
            # if the luminance value exceeds 255 after adjustion, simply add histogram bins at the upper end
            if bin > len(histogram) - 1:
                for k in range(len(histogram), ceil_bin + 1):
                    histogram.append(0)
            histogram[floor_bin] += 1 + floor_bin - bin
            histogram[ceil_bin] += ceil_bin - bin
    # Use Gausssian kernel to smooth the histogram
    histogram = gaussian_filter1d(histogram, 4)
    histogram_sum = sum(histogram)
    H = 0
    # Calculate discrete entropy
    for i in range(len(histogram)):
        p = histogram[i] / histogram_sum
        if p != 0:
            H += p * math.log(p)
    return -H


def find_parameters(cm_x, cm_y, max_distance, im):
    """
    Find a, b, c that could minimize the entropy of the image, given the
    image's cenrter of mass and the distance from the image's farthest vertex
    to center of mass
    """
    a = b = c = 0
    delta = 2
    min_H = None
    # set up a explored set to optimize running time
    explored = set()
    while delta > 1 / 256:
        initial_tup = (a, b, c)
        for parameter_tup in [(a + delta, b, c), (a - delta, b, c),
                              (a, b + delta, c), (a, b - delta, c),
                              (a, b, c + delta), (a, b, c - delta)]:
            if parameter_tup not in explored:
                explored.add(parameter_tup)
                if check_monotonically_increase(parameter_tup):
                    curr_H = calc_discrete_entropy(
                        cm_x, cm_y, max_distance, parameter_tup, im)
                    # if the entropy is lower than current minimum, set parameters to current ones
                    if min_H is None or curr_H < min_H:
                        min_H = curr_H
                        a, b, c = parameter_tup
        # if the current parameters minimize the entropy with the current delta, reduce the delta
        if initial_tup == (a, b, c):
            delta /= 2
    return a, b, c


def vignetting_correction(im):
    """
    Correct the vignetting of the image with the parameters that could minimize
    the discrete entropy.
    """
    # convert rgb image to grayscale
    imgray = cv2.transform(im, np.array([[0.2126, 0.7152, 0.0722]]))
    row, col = imgray.shape
    # calculate center of mass of the picture
    sumj = 0.0
    sumi = 0.0
    sumg = 0.0
    for i in range(row):
        for j in range(col):
            sumj += j*imgray[i,j]
            sumi += i*imgray[i,j]
            sumg += imgray[i,j]

    cm_x = sumj/sumg
    cm_y = sumi/sumg

    max_distance = math.sqrt(max(
        (vertex[0] - cm_x)**2 + (vertex[1] - cm_y)**2 for vertex in [[0, 0], [0, row], [col, 0], [col, row]]))
    # if the size of the image is too large, use the reduce-size image to get parameters and apply them on the initial image to save runnin
    if col > 500:
        ratio = col / 500
        imgray_sm = cv2.resize(imgray, (500, round(row / ratio)))
        a, b, c = find_parameters(
            cm_x / ratio, cm_y / ratio, max_distance / ratio, imgray_sm)
    else:
        a, b, c = find_parameters(cm_x, cm_y, max_distance, imgray)
    # modify the original image
    for i in range(col):
        for j in range(row):
            distance = math.sqrt((i - cm_x)**2 + (j - cm_y)**2)
            r = distance / max_distance
            g = 1 + a * r**2 + b * r**4 + c * r**6

            for k in range(3):
                modified = im[j, i][k] * g
                # if the brightness after modification is greater than 255, then set the brightness to 255
                if modified > 255:
                    modified = 255
                im[j, i][k] = modified
# read the file path from the command line
filename = "vigneet.jpg"
im = cv2.imread(filename)
cv2.imshow("input image",im)
vignetting_correction(im)
cv2.imshow("result images", im)

while 1:
 key = cv2.waitKey(1)
 if key>0:
   break
cv2.destroyAllWindows()
```

- 输入图片：



- 输出结果



- 总结：从实验的结果来看，图片四周的暗角全部消除，并且整体的图片亮度有所提升。