CT 系统参数标定及成像

摘要

关键字： 图像重建　投影

一、 问题重述

(1) 为该市中心城区 A 各交巡警服务平台分配管辖范围，使其在所管辖的范围内出现突发事件时，尽量能在 3 分钟内有交巡警到达事发地。

二、 问题分析

三、 模型假设

四、 符号说明

五、 建立模型与求解

六、 模型的评价及改进

七、 参考文献

附录 A

1.1 apps.py

```
from xlrd import open_workbook
from xlrd import XL_CELL_TEXT, XL_CELL_NUMBER, XL_CELL_DATE, XL_CELL_BOOLEAN
import numpy as np
import math
from PIL import Image

def sheet_to_array(
      filename,
      sheet_number,
      first_col=0,
      last_col=None,
      header=True):
   """Return a floating-point numpy array from sheet in an Excel spreadsheet.

   Notes:
   0. The array is empty by default; and any non-numeric data in the sheet will
     be skipped.
   1. If first_col is 0 and last_col is None, then all columns will be used,
   2. If header is True, only one header row is assumed.
   3. All rows are loaded.
   """
   DEBUG = False
   # sheet
   book = open_workbook(filename)
   sheet0 = book.sheet_by_index(sheet_number)
   rows = sheet0.nrows
   # cols
   if not last_col:
      last_col = sheet0.ncols
   if first_col > last_col:
      raise Exception("First column must be smaller than last column!")
   cols = [col for col in range(first_col, last_col + 1)]
   # rows
   skip = 0
   if header:
      skip = 1
   data = np.empty([len(cols), rows - skip])

   for row in range(skip, sheet0.nrows):
      row_values = sheet0.row(row)
      for col, cell in enumerate(row_values):
         if DEBUG and row < 2:
```

```
            print(row, col, cell.ctype, cell.value, '\n')
         if col in cols and cell.ctype == XL_CELL_NUMBER:
            data[col - first_col, row - skip] = cell.value
   return np.transpose(data)

def mat2img(mat):
   return Image.fromarray((mat / np.amax(mat) * 255).astype(np.uint8))

def img2mat(img):
   return np.array(img)

def find_mid_index(value, li):
   first_index = li.index(value)
   last_index = first_index + 1
   n = len(li)
   while(li[last_index] == value and last_index < n):
      last_index += 1
   return int((last_index + first_index) / 2)
```

1.2 init.py

```
import apps
import numpy as np

FILE_NAME = './A_attachment.xls'

sheet_01 = apps.sheet_to_array(
   FILE_NAME,
   0,
   first_col=0,
   last_col=255,
   header=False)

sheet_02 = apps.sheet_to_array(
   FILE_NAME,
   1,
   first_col=0,
   last_col=179,
   header=False)

sheet_03 = apps.sheet_to_array(
   FILE_NAME,
   2,
   first_col=0,
   last_col=179,
```

```
   header=False)

sheet_04 = apps.sheet_to_array(
   FILE_NAME,
   3,
   first_col=0,
   last_col=1,
   header=False)

sheet_05 = apps.sheet_to_array(
   FILE_NAME,
   4,
   first_col=0,
   last_col=179,
   header=False)

# test = apps.mat2img(sheet_02)
# test.save('out.jpg')

# Save csvs
np.savetxt('outputs/sheet_02.csv', sheet_02, delimiter=',')
np.savetxt('outputs/sheet_03.csv', sheet_03, delimiter=',')
np.savetxt('outputs/sheet_05.csv', sheet_05, delimiter=',')
```

1.3 std.py

```
from init import sheet_01
import numpy as np
import apps
from PIL import Image

res = 1

board = np.zeros([180, 512 * res])

offset_x = 0
offset_y = 0
sq = np.zeros([256, 256])
sq[offset_x:offset_x+256, offset_y:offset_y+256] = sheet_01
tmp_img = apps.mat2img(sq).resize([512 * res, 512 * res])
tmp_img.save('test.png')

tmp_img = tmp_img.rotate(-90)
for j in range(180):
 tmp_img = tmp_img.rotate(1)
```

```
  tmp_mat = apps.img2mat(tmp_img)
  board[j] = tmp_mat.sum(1)

img = apps.mat2img(np.transpose(board)).resize([180, 512])
img.save('outputs/std.jpg')
```

1.4 ellipse.py

```
from init import sheet_02
import numpy as np
import apps
from PIL import Image

# data = sheet_02.copy()
# act_img = apps.mat2img(diff)
# act_img.save('outputs/act.jpg')

# find the ellipse
vexs = []
for i in range(180):
  tmp = []
  for j in range(1, 512):
    if (sheet_02[j, i] != 0 and sheet_02[j-1, i] == 0):
      tmp.append(j)
    elif (sheet_02[j, i] == 0 and sheet_02[j-1, i] != 0):
      tmp.append(j-1)
  if (len(tmp)>2):
    if (tmp[1] - tmp[0] > tmp[3] - tmp[2]):
      tmp.remove(tmp[3])
      tmp.remove(tmp[2])
    else:
      tmp.remove(tmp[0])
      tmp.remove(tmp[0])
  vexs.append(list(tmp))

# get the length
length = []
for item in vexs:
  length.append((item[1] - item[0]))

max_axis = max(length)
min_axis = min(length)

max_index = apps.find_mid_index(max_axis, length)
min_index = apps.find_mid_index(min_axis, length)
```

```
print(max_axis, min_axis)
print(max_index, min_index)
```

1.5 angles.py

```
import numpy as np
import apps
from PIL import Image
from math import acos, pi

RATIO = ((288 / 80) + (107 / 30)) / 2

cir_data = np.loadtxt('inputs/SplinedRound.csv', delimiter=',')
ell_data = np.loadtxt('inputs/SplinedEllipse.csv', delimiter=',')

apps.mat2img(ell_data).save('haha.png')

cir_index = []
ell_index = []
for i in range(180):
  cir_index.append(list(cir_data[:, i]).index(max(cir_data[:, i])))
  ell_index.append(list(ell_data[:, i]).index(max(ell_data[:, i])))

diff = (np.array(cir_index) - np.array(ell_index))

angles = []
for i in range(180):
  cos_value = diff[i] / 45 / RATIO / 1.01085271318
  angle_tmp = acos(cos_value) / pi * 180
  if (i == 0):
    angles.append(angle_tmp)
  elif (angle_tmp < angles[i-1]):
    angles.append(360 - angle_tmp)
  else:
    angles.append(angle_tmp)

np.savetxt('outputs/angles.csv', angles, delimiter=',')
```