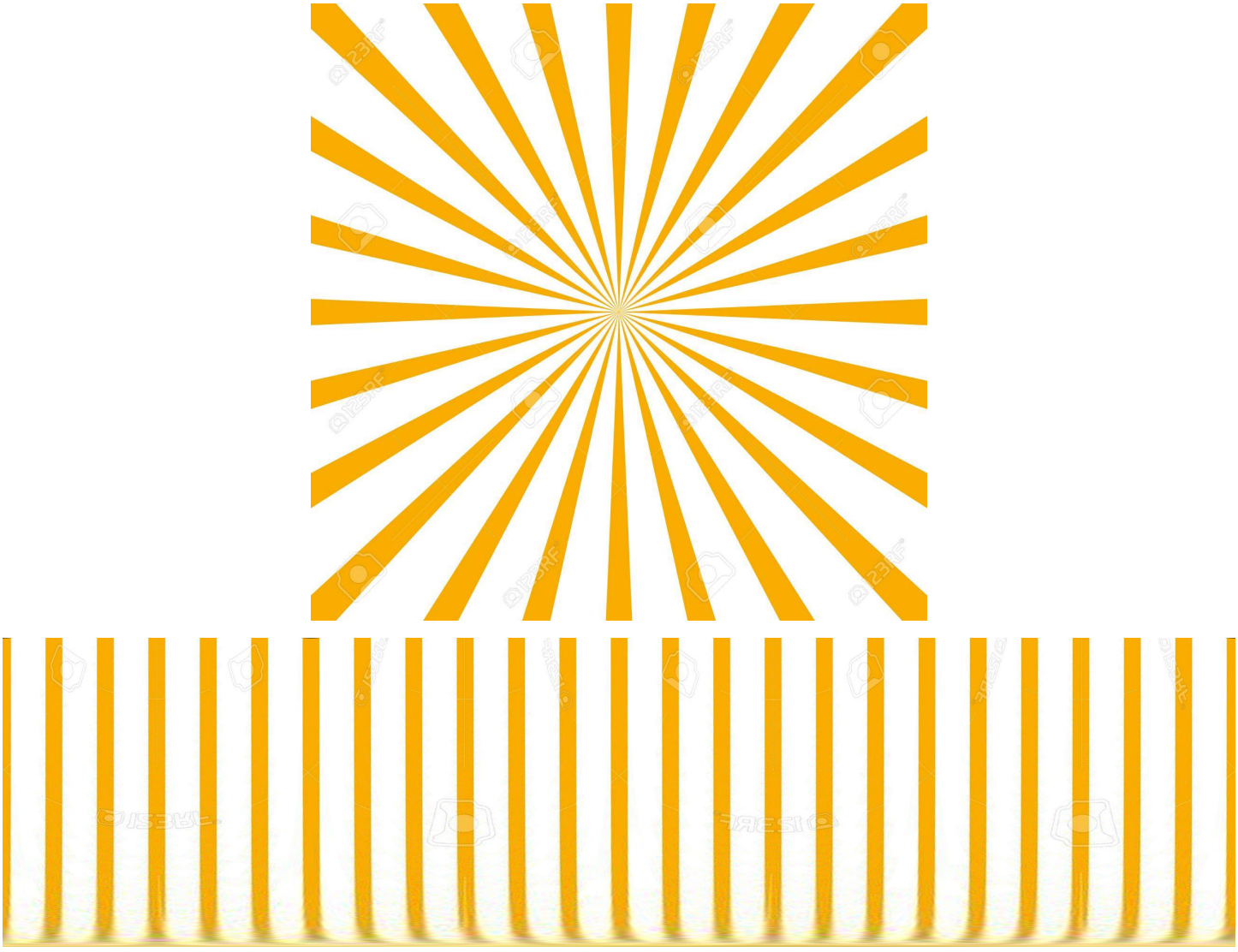
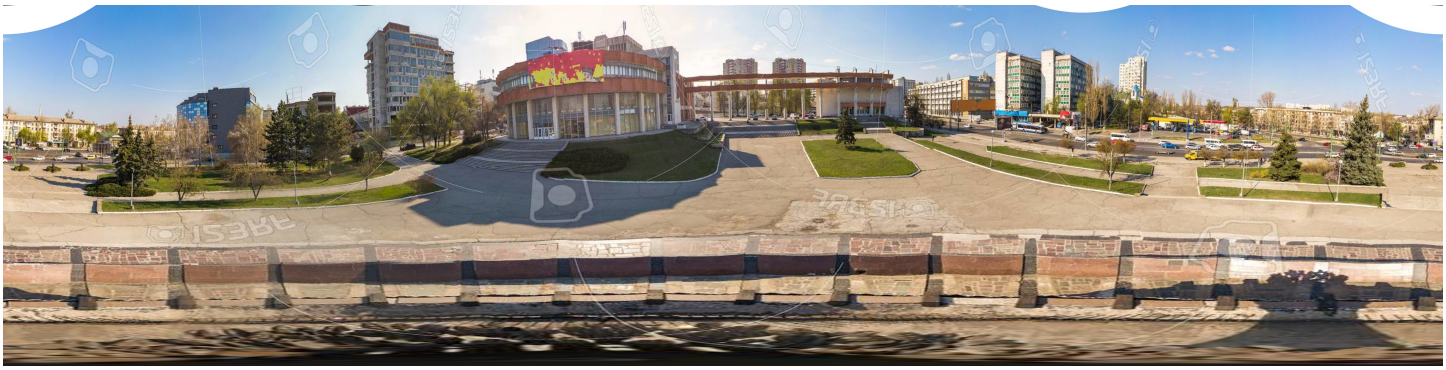
Radial Image Processing

Updated: February 10, 2020

Started 12/17/2019





```
import cv2
from unwrapper import SphereUnwrapper
import sys

def unwrap(filename):
  cv2.namedWindow("preview")
  capture = cv2.VideoCapture(filename)
  _, frame = capture.read()
  unwrapper = SphereUnwrapper.makeFromSize(frame.shape[0])

  unwrappedFrame = unwrapper.unwrap(frame)
  cv2.imshow("", unwrappedFrame)

  while capture.isOpened():
    _, frame = capture.read()
    if frame is not None:
      unwrappedFrame = unwrapper.unwrap(frame)
      cv2.imshow("", unwrappedFrame)

    if cv2.waitKey(1) == ord('q'):
      break

  capture.release()

unwrap(sys.argv[1])
cv2.destroyAllWindows()
cv2.waitKey(0)
```

```
import numpy
import cv2

class SphereUnwrapper:
  def __init__(self, innerRadius, outerRadius, centerX, centerY, angle, interpolation=cv2.INTER_CUBIC)
    self.interpolation = interpolation
    self.buildMap(innerRadius, outerRadius, centerX, centerY, angle)

  @classmethod
  def makeFromSize(cls, size):
    return cls(0, size/2, size/2, size/2, 0)

  def buildMap(self, innerRadius, outerRadius, centerX, centerY, angle):
```

```
    absoluteOuterRadius = centerY + outerRadius
    absoluteInnerRadius = centerY + innerRadius

    outerCircumference = 2*numpy.pi * outerRadius
    mapWidth = int(outerCircumference)
    #TODO find actual vertical FOV angle (instead of 90)
    mapHeight = int(mapWidth * (90/360))

    rMap = numpy.linspace(outerRadius, innerRadius, mapHeight)
    thetaMap = numpy.linspace(angle, angle + float(mapWidth) * 2.0 * numpy.pi, mapWidth)
    sinMap = numpy.sin(thetaMap)
    cosMap = numpy.cos(thetaMap)

    map_x = numpy.zeros((mapHeight, mapWidth), numpy.float32)
    map_y = numpy.zeros((mapHeight, mapWidth), numpy.float32)
    for y in range(0, mapHeight):
      map_x[y] = centerX + rMap[y] * sinMap
      map_y[y] = centerY + rMap[y] * cosMap
    (self.map1, self.map2) = cv2.convertMaps(map_x, map_y, cv2.CV_16SC2)

  def unwrap(self, img):
    output = cv2.remap(img, self.map1, self.map2, self.interpolation)
    return output
```