University of Freiburg
Faculty of Environment and Natural Resources
Module: GIS Plus
Lecturer: João Paulo Pereira
Module Coordinator: Holger Weinacker
Authors: Luka Kern, Nele Stackelberg and Felix Rentschler

July 6, 2018

# Project 06: Rasterizer Function

## 1 Aim of the Project

The aim of this project was to create a rasterizing tool that generates a raster-file from a given shape-file. The idea of such a rasterizing tool is to load the geometries of a shape-file into a geometry collection, implement a regular grid within the bounding box of this geometry collection and assign a value to each grid cell. For the value assignment it's either possible to use attribute values of the geometry or to use a binary format depending on the presence of a geometry.

### Approach

To approach this task first of all we developed a script which randomly generates geometries – polygons, points and linestrings – and saves them as shapefiles. Those shapefiles served as test data. As the rasterizing tool itself we built a function with several options for the user whereby the filepath to a shapefile is the essential argument. The function works according the following pattern: The geometries of the given shapefile are loaded into a geometry collection. Then the bounding box around this collection is defined with an additional frame. Within this frame a regular grid is generated as well as point geometries representing each grid cell. With the coordinates of the generated point geometries it is possible to run geometric queries testing if a positive value should be assigned to the pixel. Regarding the value assignment the function is using a binary format with a highlighting of ovelapping geometries. Finally the raster can be saved in a 8bit format.

## Structure of the Function

The path of the shapefile is the functions first argument. A further argument is the amount of pixels, which is used as a proxy for the resolution. The buffersize in pixels and the name of the output file can additionally be defined as arguments for the function.
Furthermore, the user has the opportunity to specify the name of the result file, to preview the result inline, as well as suppressing the saving of the tiff file. For the case, that no shapefile is available to run the function, the user can generate random geometries with the provided script (see above).

### Load in the Shapefile

To load in the shapefile, the *Fiona* package is used. Every geometry object of the shapefile is stored in a GeometryCollection from the *Shapely* package.

## Calculate the Resolution

To calculate the resolution, the minimum and maximum dimension of the geometry collection on the x- and y- axes are extracted. Those values are used to determine the range dimension.
The resolution depends on the amount of pixels in each dimension. Because the amount of pixels is one dimension, that is defined as input in the function, the range is divided by this value to get the resolution. Because the x- and y- range may differ, the mean of those two values is used in the calculation above.

## Defining the Bounding Box and create a Grid

We create a bounding box around the spatial dimension of the shapefile. The purpose of this step is to present the objects in the resulting image with distance to the margin.
The functions argument "buffer" describes the amount of pixels in the bounding box. To transform the pixel value to the coordinates, it is first multiplied with the resolution, the result is the width of the boundary box.
The dimension of the geometry collection (xmin, ymin, xmax, ymax) is used to define the dimension of the outer frame of the bounding box by adding and subtracting the boundary box width.
The minimum and maximum coordinate values (dimension) of the outer frame of the boundary box are used to create a grid with the function *numpy.mgrid*. The coordinates of every grid-cell (= every pixel) are saved as point-geometry. Those geometries are stored in a list called geom_pixels.

## Within Query

Now, that the position of every grid-cell is known, the vector geometries can be transferred to the raster-grid. The query, that defines wheter a geometry object covers a pixel (metaphorically speaking: imagine a polygon laying on a squared paper and transforming it manually) depends on the geometry type of the vector file. If it is a Point, the query asks if the pixels position (from geom_pixels) equals the points position (+- 0.5*resolution in both x- and y-dimension) in the vector data. The addition respectively subtraction of 0.5 times the resolution makes sure that every point is detected, even if it is not on the midpoint of the grid cell. If the vector geometry is a line, a buffer around the line is calculated. The within-function of the shapely-package is than used to query if a point is in this buffer. The same query is also used for polygon-geometries, where the within-function is applied directly to the polygon and not to a buffer. The query is applied to all geometries of the geometry collection. For each geometry a list with boolean values is the result. These list are appended one after another as sublists.

## Writing the result in an array

In the next step the sublists are combined. Therefore the sum off all sublist is written to a new list. The sum of boolean values is the amount of "True"-values. To create an array of the right size out of this list, it has first to be transformed into sublist, each with the length of the amount of pixels on the x-axis. The resulting format can be transformed to a numpy-array.

## Set Radiometric Resolution

To make sure that all values of the array will be displayed in the 8bit TIFF file, the values are transformed to the range between 0 and 255. The calculation of every value $x_{i\_new}$ from its former value $x_i$ in the grid $x$ is computed according to the following formula: $x_{i\_new} = 255 * \frac{x_i}{x_{max}}$

### Flip Array and Save as TIFF

If the array was plotted at this stage, pythons characteristics to show the y-achsis the reverse way would result in a inverted image. Therefore the array is flipped the up/down direction with the flipud-function of the package numpy. After all this has been computed, the result can be saved as TIFF-file. For this purpose the scikit-package provieds the imsave-function, which is used here.

## 2 Result

## 3 Issues

### Solved Issues

To provide a general function, that can be applied to a shapefile, no matter if the geometry type was "Points", "Lines" or "Polygons", those three possibilities had to be covered. The Solution of this Issue should have become clear in the section 1 ("Within query).

To set the functions argument "resolution" instead of the "pixels" would have been a more intuitive value. The explanation why we did not use this argument is this: The runtime of the function mainly depends on the raster files resolution. To set an appropriate resolution, the user has to know the range of coordinate-values (and type of coordinates) in the shapefile and divide it by the amount of pixels. For raster-images with increasing pixels in one dimension, the runtime increases exponentially. With more than approximately 200 pixels the runtime will quickly exceed one minute (on an average laptop). Because during development the function was applied to shapefiles, whose dimensions and coordinates, the approximate amount of pixels seemed to be the more convenient argument.

To make the group work as efficient as possible, especially as not all participants were able to be in Freiburg, the platform github was chosen to facilitate the workflow. The repository `https://github.com/lukakern/GisPlus.git` was created for the project. Since working with git in the console or using the software pyCharm was new to all group members, there were some difficulties in handling the files by merging, pushing and pulling. It took time until these issues were solved, but it was worth it, especially since these experiences will be very helpful for future projects.

### Issues in Progress

As a result the function creates raster files where the presence or absence of the shapefiles geometries is displayed and overlapping geometries are represented by lighter grey values. This is a possibility that works for all given shapefiles. However the representation of attributes in the raster file by different gray values or even raster bands would ensure a broader application of the function.

Different users may also want to be able to choose from different radiometric resolutions for the output TIFF file instead of the hard-coded 8bis resolution in the function. Therefore it would be nice to have the resolution as an argument for the function.

The function saves the TIFF file without reference system. This is probably a problem for many users, as the image cannot be processed further as easily in standard GIS-software. To include this feature to the function the reference-system (EPSG-number) of the input-shapefile has to be accessed in the beginning of the function and reused when writing the TIFF-file.