Writing Image Processing Algorithms using the Python Raster Function

Gregory Brunner  Jamie Drisdelle  Feroz Abdul Kadar
Agenda

- Introduction
- Anatomy of a raster function
- Building raster models
- Additional considerations
- Q&A
Introduction

Raster Functions
What’s a Raster Function?

The Fundamentals

- Mapping of one raster to another.
- Loosely, \[ I : \mathbb{N}^3 \rightarrow \mathbb{R} \quad F : (I, \Phi) \rightarrow \mathbb{R} \]
- On-demand. Transient.
- Different from a geoprocessing tool.

... a transformation of one raster into another.
Chaining Raster Functions

Raster Model

A raster model encapsulates your algorithm as a tree of functions.
Chaining Raster Functions

An example: Landsat 8 Pansharpened
Demo

Apply a raster functions to image layers in ArcMap

Browsing raster models on an image service in ArcGIS.com
Python Raster Functions
What’s a *Python* Raster Function?

The Extended Model

- Transforming rasters—image processing and analytic algorithms—in Python.

- Implement a raster function from the comfort of your *Python module*.

- *Architecture:* Module loaded by an adapter—Python-aware *and* a first-class participant in the function chain.

*Your Python module—assisted by ArcGIS—is a raster function.*
What’s a *Python* Raster Function?

The Extended Model

- Function Raster Dataset
- Mosaic Dataset Item
- Mosaic Dataset
- Image Service
Motivation

Why Raster Functions in Python?

- Extend ArcGIS—participate in a *raster model*.
- Primary pipeline for image data in ArcGIS—*processing, analyzing, and visualizing*.
- On-the-fly at display resolution, or in batch at full resolution.

**Why Python?**

- Friendly & easy to learn. “readability first”. “batteries included”.
- Huge collection of libraries. Vibrant community of *Pythonistas* and *Pythoneers*.
- “...*de facto* superglue language for *modern scientific computing*.”
- “...tools for almost every aspect of scientific computing are *readily available in Python*.”
The API

How do I create Raster Function in Python?

- How does ArcGIS Desktop or Server *interact* with my raster function?
  - Get started—step-by-step guide
  - Real-world or reference implementations—excellent springboard
  - Well-documented API reference

- What additional libraries are needed? How complicated is it?
  - Lightweight design—no external dependencies outside of NumPy to begin with.
  - ArcGIS’ *adapter* provides assistance—opt out to take control of specific aspects.

*Create a new raster function using simple and familiar Pythonesque constructs.*
Hello, World!

```python
import numpy as np

class HelloWorld():
    def __init__(self):
        self.name = "Hello World Function"

    def getParameterInfo(self):
        return [{
            'name': 'r',
            'dataType': 'raster'
        }]

    def updatePixels(self, tlc, shape, props, **pixelBlocks):
        r = pixelBlocks['r_pixels'] + 10
        pixelBlocks['output_pixels'] = r.astype(props['pixelType'])
        return pixelBlocks
```
The API

__init__

- customize our function object—a specific instance of our class—as soon as it's created.

- Define raster function *name* & *description*.
The API

getParameterInfo()

- Define all *input parameters* to the function.

- For each parameter, define:
  - Name (Identifier)
  - Display Name
  - Long Description
  - Data Type
  - Default Value
  - Required vs Optional
  - ...

![Image of parameter information dialog box](Image)
The API

cdef getConfiguration():

    · How are input rasters read—Padding, Mask, ...?

    · How's the output raster constructed—inherit NoData, Metadata, ...?

    · **Given:** scalar (non-raster) values.

    · **Returns:** dictionary containing configuration attribute values.
The API

selectRasters()

- Define a subset of input rasters.
- Pixels read from selected rasters.
- **Given:** properties of the requested pixel block, all scalar parameter values.
- **Returns:** names of selected rasters.
The API

`updateRasterInfo()`

- Defines the output raster.
- Invoked each time a dataset containing the Python raster function is initialized.

Given: Raster info associated with all input rasters.

Returns: Raster Info of the output raster.
**The API**

updatePixels()

- **Workhorse of the raster function. Process Pixels.**

- **Given:**
  - Expected pixel-block size+location
  - output raster properties (map space)
  - pixels+mask of selected input rasters

- **Returns:** Pixels+mask of requested pixel block.
The API

`updateKeyMetadata()`

- Create or update dataset- or band-level metadata.

**Given:**
- property names
- band index
- current key metadata values

**Returns:** updated values of given properties
The API

isLicensed()

• **Given:**
  • info on parent product,
  • context of execution.

• **Returns:**
  • OK to Run (Boolean)—Licensed to execute or not?
  • Expected product level and extension.
Compound Topographic Index
Compound Topographic Index (CTI)
A steady state wetness index, a.k.a. Topographic Wetness index (TWI)

T.R. Matthews et al. 2015
There’s a need in the agricultural and natural resources community

To explore Python Raster Functions

To understand the limits of a Python Raster Function
  - Can I use SciPy?
  - Does it make sense to use for loops?
  - How does a Python raster function chain with other raster functions?
Compound Topographic Index

Calculations

\[
CTI = \ln \left( \left( \text{flow}_{\text{accumulation}} + 1 \right) \times \frac{\text{cellsize}}{\tan(\text{slope})} \right)
\]
Compound Topographic Index

Calculations

DEM → Slope Function → Flow Accumulation → CTI Function → CTI

Python powered
CTI: Code

c_slope(slope_deg):
  pe = slope_deg * pi/180  # np.arctan
  urn slope

c_cti(slope, flow_acc, cellsize):
  _slope = np.tan(slope)
  _slope[tan_slope==0]=0.0001
  = np.log(((flow_acc+1)*cellsize)/
  urn cti
Site Suitability Analysis
Site Suitability Analysis

Weather Variables

Remap Table

<table>
<thead>
<tr>
<th>Operation</th>
<th>Variable</th>
<th>From</th>
<th>To</th>
<th>Remap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air Defense</td>
<td>Temperature</td>
<td>50</td>
<td>100</td>
<td>-2</td>
</tr>
<tr>
<td>Air Defense</td>
<td>Cloud Ceiling</td>
<td>2460</td>
<td>4920</td>
<td>-1</td>
</tr>
<tr>
<td>Air Defense</td>
<td>Wind Speed</td>
<td>0</td>
<td>36</td>
<td>0</td>
</tr>
</tbody>
</table>

Output Function Raster

Legend:
- Very bad
- Bad
- Average
- Good
- Very good
Site Suitability Analysis

Weather rasters → Remap functions → Cell statistics function → Operational threshold

Python powered
Site Suitability Analysis

```python
# use zonal thresholds to update output pixels...
if ZT is not None and len(ZT.keys()):
    for k in (zoneIds if zoneIds is not None else [None]):
        T = ZT.get(k, None)  # k from z might not be in ztMap
        if not T:
            continue

        for t in T:
            I = (z == k) if z is not None else np.ones(v.shape, dtype=bool)
            if t[0] and t[1]:     # min and max are both available
                I = I & (v > t[0]) & (v < t[1])
            elif t[0]:
                I = I & (v > t[0])
            elif t[1]:
                I = I & (v < t[1])
            p[I] = (t[2] if t[2] is not None else self.defaultTarget)

pixelBlocks['output_pixels'] = p
```
Building *Raster Models*
Templatized chains of raster functions
What’s a Raster Model?

The basic concept

A templatized raster model replaces one or more inputs with place-holder Variables.
Raster Model

Basic workflows with “function template”

• Create new function templates
  • Via Raster Function Template Editor
  • Layer > Symbology > Export As Raster Function Template
  • Function Raster Dataset > Functions > Save as

• Raster Layer in a Map
  • Image Analysis Window
  • Raster Functions Pane in Pro
  • Layer > Properties > Functions tab

*A raster function template makes your processing or analysis* **portable.**
Raster Model

Advanced workflows with function templates

- **On a Mosaic Dataset**
  - Populating a mosaic using the Add Rasters tool
  - Mosaic dataset items
    - *Batch Edit Raster Functions or*
    - *Edit Raster Function Geoprocessing Tool.*
  - As *Processing Templates*

- **On an Image Service**—for server-side processing

*Learn more at [https://github.com/Esri/raster-functions#raster-function-templates](https://github.com/Esri/raster-functions#raster-function-templates).*
Raster Model

Properties of a function template

- Name & Description
- Type: *Item, Group, or Mosaic.*
- Definition Query
- *Fields that control grouping.*
Demo

Apply CTI raster model to image layers on map using ArcGIS Pro
Demo

Apply *suitability* raster model to a mosaic dataset using ArcMap

Browse processing templates of the mosaic layer
Additional Considerations
Performance

- Vectorize.
- Use NumPy and SciPy.
- Use *Cython*: For production-grade performance.
- Bridge to C/C++ implementations via *ctypes* or *Boost.Python*.

*Leverage well-known options to optimize time-critical parts of your raster function.*
Publishing & Deployment

- Python version
  - Desktop vs. Pro: Python 2.7 vs. 3.4
  - Desktop vs. Server: 32-bit vs. 64-bit Python 2.7

- Package dependencies

- Binary deployment
  - CPython bytecode. Cython compiled binary.
  - `isLicensed` method to restrict usage.
GitHub
Where do functions and templates live?

- [https://github.com/Esri/raster-functions](https://github.com/Esri/raster-functions)

- Curated collection of raster *functions* and *templates*.

- Go ahead:
  - [Browse samples](https://github.com/Esri/raster-functions) to learn more.
  - [Fork](https://github.com/Esri/raster-functions) the repo. Experiment.
  - Log defects or enhancement requests as [issues](https://github.com/Esri/raster-functions).
  - Send [pull requests](https://github.com/Esri/raster-functions) to contribute code.

*GitHub enables collaboration.*
Wiki

Where do I find the story?

- https://github.com/Esri/raster-functions/wiki

- Details of interaction between ArcGIS and your Python raster function.

- Recommendations and techniques for writing effective raster functions

- Anatomy of a Python Raster Function
  - getParameterInfo
  - getConfiguration
  - updateRasterInfo
  - selectRasters
  - updatePixels
  - updateKeyMetadata
  - isLicensed

- Writing Effective Raster Functions
  - Performance Considerations
  - Handling NoData
  - Key Metadata
  - Aggregation and Grouping
  - Using Cython
  - Debugging
  - Deployment
In Closing

- Powerful pipeline for processing, analysis, and visualization.
  - *Raster models* built using *raster functions* for on-demand processing.
  - Build dynamically derived information products.
  - Manage redundancy, provenance/authenticity, reproducibility.

- Using simple constructs in Python: participate and exploit.

- Code and docs are on [GitHub](https://github).

- We’ll help you along the way—[@gbrunner](https://twitter.com/gbrunner), [@jdrisdelle](https://twitter.com/jdrisdelle), and [@akferez](https://twitter.com).
Questions?
Rate This Session
Use the *Esri Events* mobile app

**Sessions on related topics:**

- *Change Detection using the Python Raster Function*—Tuesday at 1:00
- *Getting Started with the JS API for Multidimensional Image Services*—Tuesday at 1:00
- *Automating Image Management and Dissemination using Python*—Wednesday at 10:30
- *Developing Applications which use Oblique Imagery in ArcGIS*—Wednesday at 4:00
- *Getting Started with Imagery using the Web AppBuilder for ArcGIS*—Wednesday at 4:35

[http://www.esri.com/events/devsummit/agenda/detailed-agenda](http://www.esri.com/events/devsummit/agenda/detailed-agenda)