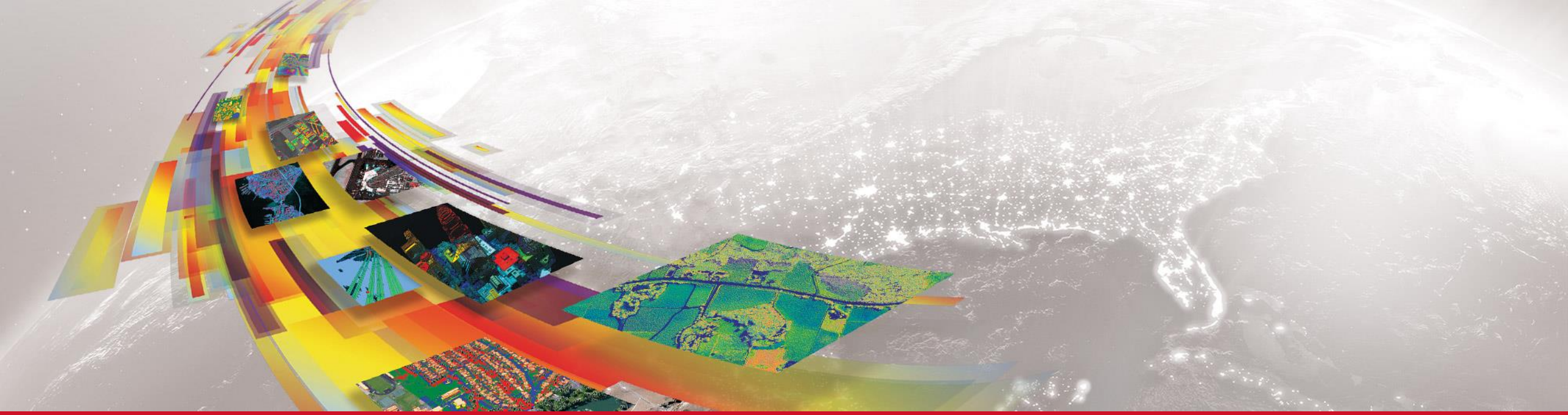


OPEN SOURCE TECHNOLOGIES FOR ENVI AND IDL

ZACHARY NORMAN

Senior Solutions Engineer

December 18th, 2018

HARRIS® TECHNOLOGY TO CONNECT, INFORM AND PROTECT™

Today's Speakers

Zachary Norman

Solutions Engineer

znorman@harris.com

Agenda

Introduction

What is Open Source?

IDL Python Bridge

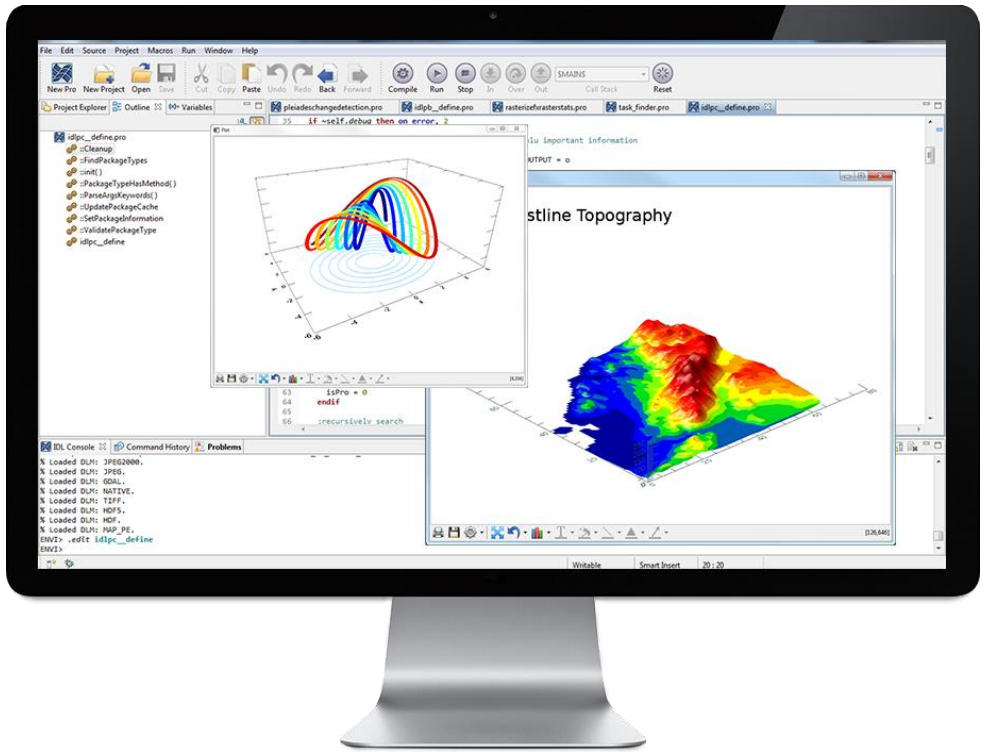
IDL Package Manager

GitHub

Upcoming Packages to GitHub

Demo!

Questions



Harris Geospatial Solutions

Geospatial Solutions and Services

Harris' innovative mapping and visualization capabilities are designed to meet the mission requirements of federal and civil government as well as a broad range of commercial users. Our integrated imagery analytics, cutting edge sensor types and scalable data production and data management offerings allow the user to seamlessly gain situational awareness using advanced, easy to use, remote sensing technology.

ENVI Analytics

Deep Learning

Geiger-mode LiDAR

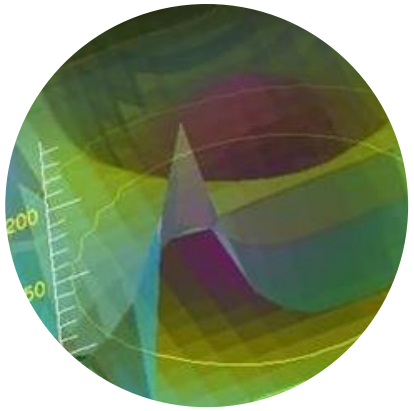
Real-time AIS Tracking

Data Production and Services

Data Brokerage Services

HARRIS®

Geospatial Portfolio



IDL

ENVI

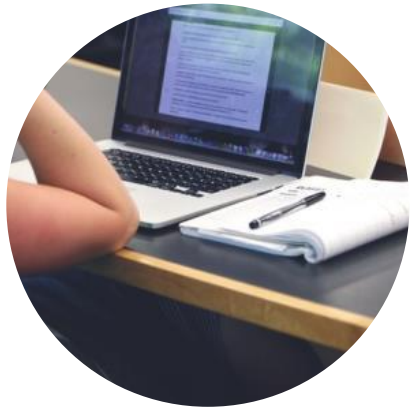
SARscape

Geospatial Services Framework

Jagwire

MapMart Marketplace

Deep Learning



Training & Consulting

What is Open Source?

Open Source Matters

Great for community and user driven development

Power is in the hands of users

Want a change, you can make it!

IDL Integration for an Open Source World

IDL Package Manager (IPM) makes it easier for developers to share IDL code in open source ecosystems

- Create, install, update, and remove IDL packages, which are zipped files & folders
- Packages can contain IDL pro code, IDL save files, and/or IDL DLMs
- New IDL_PACKAGE_PATH preference, and the IDL path is automatically updated

IPM-ready libraries are now available

```
ipm, /install, 'https://github.com/csalvaggio/IDL_RIT_Salvaggio'
ipm, /install, 'https://github.com/hadfieldnz/idl-motley'
ipm, /install, 'https://github.com/hadfieldnz/idl-roms'
ipm, /install, 'https://github.com/mankoff/kdm-idl'
ipm, /install, 'http://packages.idldev.com/idldoc.zip'
ipm, /install, 'http://packages.idldev.com/mgunit.zip'
```

IDL-Python Bridge

Bi-directional bridge lets you easily run IDL routines or ENVI analytics from your language of choice

Can also use ENVI Py to call ENVI Tasks

IDL-Python Bridge

```python
from idlpy import IDL
import numpy as np
import os

#start ENVI
e = IDL.envi(HEADLESS = 1)

# get task definition from IDL
task = IDL.ENVITask("BuildMosaicRaster")
task.INPUT_RASTERS = rasters
task.RESAMPLING = 'Nearest Neighbor'
task.FEATHERING_METHOD = 'edge'
task.OUTPUT_RASTER_URI = e.GetTemporaryFilename()
task.execute()
```

ENVI Py Directly

```python
from envipyengine import Engine
from envipyarc import GPToolbox
import arcpy

# make toolbox
engine = Engine('ENVI')
task_list = [engine.task('SpectralIndex')]
envi_toolbox = GPToolbox(task_list)
toolbox_file = envi_toolbox.create_toolbox('c:\\my_envi_tools')
arcpy.ImportToolbox(toolbox_file)

# run the toolbox
input_raster = 'C:/Program Files/Harris/ENVI54/data/qb_boulder_msi'
index = 'Normalized Difference Vegetation Index'
result = arcpy.SpectralIndex_envi(input_raster,index)
print(result)
```

Python Bridge Examples

Use IDL to create a Scikit-learn classifier:

```
; Read data
read_seeds_example_data, data, labels, $
  N_ATTRIBUTES=nAttributes, N_EXAMPLES=nExamples, $
  N_LABELS=nLabels, UNIQUE_LABELS=uniqueLabels

; Import scikit-learn and our classifier
>>>import sklearn
>>>from sklearn.ensemble import RandomForestClassifier

; Create a random forest classifier
clf = python.RandomForestClassifier(n_estimators = 25, max_depth = 5, $
  min_samples_split = 2, random_state = 42)

; fit to the data - transpose because python os row major, not column major
clf = clf.fit(transpose(data), labels)

; Classify the first example
print, clf.predict(data[*,0]), /IMPLIED_PRINT

; specify a file to save the model to
tempFile = 'C:\users\username\mymodel.p'

; import the pickle module
pickle = python.import('pickle')

; Save the model for later with the latest, most efficient protocol
!NULL = pickle.dump(clf, python.open(tempfile, 'wb'), protocol=-1)
```

Use ENVI from Python to read data for Keras:

```
import * from idlpy
import numpy as np
import keras

# start ENVI **must be headless**
e = IDL.envi(HEADLESS = 1)

# load a previously saved model
modelFile = 'C:\\users\\username\\mymodel.h5'

# load the model
clf = keras.models.load_model(modelFile)

# read some data from a raster
raster = e.openRaster('C:\\users\\username\\myraster.dat')
dat = np.transpose(raster.getData())

# classify the data
# TODO: extract chips first then predict in batches
classified = clf.predict(data)

# create output raster and save result
outRaster = IDL.ENVIRaster(np.transpose(classified),
  URI = 'C:\\users\\username\\myraster.dat')
outRaster.save()
```

GitHub and Git

GitHub (https://github.com/):

- A development platform for hosting git repositories
- Free for open-source projects
- Extensive userbase
- More than just source control

Git (https://git-scm.com/):

- A tool for source control of files, most often code
- Track changes, merge branches of code, and makes examining previous work easy
- CLI or UI

ENVI + IDL on GitHub

We are on GitHub!

ENVI and IDL GitHub organizations:

- https://github.com/envi-idl
- https://github.com/interactive-data-language

Repositories vs. Packages and Concepts

For this webinar, “IDL Packages” are synonymous with GitHub repositories

GitHub repositories are meant for source code, but you can store other types of files there

Releases are snapshots of a repository and releases can have additional files attached (i.e. compiled source code)

When anyone can contribute to an open source, IDL package, how do you verify code integrity and usability?

Solution: IDL Package Creator

One-stop-shop to ensure code integrity by providing a simple platform for:

- Writing and running unit tests
- Generating documentation
- Compiling source code

Inspired by npm, yarn, and the Angular CLI

Open sourced (coming soon) for community-driven development

Built around GitHub's rest API

IDL Package Creator: About

Two types of packages: global and local

Local packages are for development and advanced users

Global packages are for deployment

Object based API and CLI for executing commands

```
; open our global package
c = idlpc()

; open a local package
c = idlpc('C:\some\folder\on\disk\my-package')

; shortcut to open a local package if on search path
c = idlpc('my-package')
```

```
C:\>ipc --global remove uavtoolkit
Licensed for use by: Harris Geospatial Tradeshow
License: MNT-5507041:****-****-****-D10E
License expires  4-Jan-2019.
Using global installation location
WARNING: Entry point not specified in idl.package.json file
  Processing dependencies
Removed package "github-uavtoolkit"
C:\>ipc --global add uavtoolkit
Licensed for use by: Harris Geospatial Tradeshow
License: MNT-5507041:****-****-****-D10E
License expires  4-Jan-2019.
Using global installation location
  Retrieving package information for "envi-idl/UAVToolkit"
    Resolving dependencies...
  Installed package "github-uavtoolkit" with release tag "^v2.3.3"
```

IDL Package Creator: Testing with Sol and Luna

Unit tests are critical for writing dependable code

Sol: Test runner, wrapped by the IDL Package Creator

Luna: Test writer for individual files

Code coverage and profiling

Inspired by Jasmine and Karma

```
compile_opt idl2

; main level program (Luna can only be instantiated in a main level program)
l = luna(/PROFILE)

  ;create a test suite
  s = l.suite('Sample test suite which has a few tests with' )

    ;create a test for our suite
    it = s.test('a test with expectations that pass')

      ;make sure our routine runs
      (it.expects('example')).toRunProcedure

; summarize our tests
l.generateTestSummary
end
```

IDL Package Creator: Documenting

Existing solution is [idldoc]

Based on markdown, uses [pandoc]

Used for generating the courses for ENVI + IDL

Live example: https://envi-idl.github.io/UAVToolkit/

```
;+
; :Examples:
;    See ![FILE:./README.md] for examples of how to use this routine.
;
; :Description:
;    Simple routine that can be used to perform input validation
;    for any routine to ensure that parameters are present or a
;    certain type. You can specify each possible type that the
;    data value can represent and optionally require that the value have
;    all of the types. Here are the generic types that can be used:
;
;    - required : If set, the value must be present and defined.
;
;    - array    : If set, value supplied must be an array.
;
;    - number   : If set, value supplied must be a number.
;
;    - file     : If set, value must be a file on disk.
;
;    - directory: If set, value must be a folder on disk.
;
;    This routine uses IDL's `isa` function to make the comparison so,
;    in addition to the types above, you can specify anything else that
;    can pass as an argument. Some exampled are: byte, int, long, float,
;    hash, orderedhash, enviraster. They can be any IDL-specific data
;    type and it can also be the type of object such as idlgrwindow or
;    any named, custom object type.
;
; :Params:
```

AwesomeLabelRegions

This task uses IDL's `label_region()` on images in a tiled fashion. Because this uses `label_region()` it is focues on processing binary images with pixels values that are zero or nonzero. The out-of-the-box LabelRegions algorithm works with images that are non-binary (i.e. classification images) and, understanding that, this task is for a specific use case: binary datasets.

For binary images, this routine is about 2x faster than the LabelRegions task. For a 120 MB binary mask with 10980 columns and rows here are performance metrics:

COPY

```
AwesomeLabelRegions : 8.6 seconds
LabelRegions        : 16 seconds
```

Here is an example of how you can run this routine:

COPY

```
; Start the application
e = envi()

; load our awesome algorithms
awesomeENVIAlgorithms, /INIT

; Open an input file
File = filepath('qb_boulder_msi', SUBDIR = ['data'], $
  ROOT_DIR = e.ROOT_DIR)
```

IDL Package Creator: Building

Compiling source code is not hard to do, but repetitive

Can be a hassle to do over and over again

Will automatically run unit tests before building

Resolves all dependencies into a single entry point

```
IDL> c.build, /NO_TEST
WARNING: Tests skipped per user request
Cleaning up existing folder...
Building package in "C:\Users\Traininglead\Documents\github\Tasks\SE-AwesomeENVIAlgorithms\dist" with types:
  idl

Building type "idl"...
  Searching for files to copy and build...
  Found files to process...
  Initializing child process...
  Copying and building files, may take a minute or two...
    Resolving dependencies
    Dependencies resolved in:
      C:\Users\Traininglead\Documents\github\Tasks\SE-AwesomeENVIAlgorithms\dist\awesomeenvialgorithms.sav
  Copying and building additional files, may take a minute or two...
  Zipping contents
  Cleaning up

  Successfully built package!
```

Packages on GitHub

Upcoming:

- IDLPackageCreator
- AwesomeENVITools
- AwesomeENVIAlgorithms
- MaskingToolkit

Existing Packages:

- UAVToolkit
- FXExtras

IDL - C:\Users\Traininglead\Documents\github\Tools\SE-IDLPackageCreator\src\idlpd\idlpd__define.pro - IDL

File Edit Source Project Macros Run Window Help

New Pro New Project Open Save Cut Copy Paste Undo Redo Back Forward Compile Run Stop In Over Out $MAIN$

Project Ex... Outline Variables

- idlpd__define.pro
  - idlpd_process_md_file( )
  - idlpd_process_txt_file( )
  - idlpd_process_block
  - idlpd_blocks_to_strings( )
  - idlpd_extract_blocks( )
  - ::init( )
  - ::document
  - idlpd__define

idlpc__define.pro | idlpb__define.pro | idlpd__define.pro

```
920       endelse
921
922       ;check if we have parameter syntax
923       hadParams = 0
924       case (1) of
925         blockHash.hasKey('%params'):begin
926           if ~(types[i] eq 'function') then syntaxCall += ', ' ; check if we need comma for procedure
927           syntaxCall += blockhash['%params']
928           blockHash.Remove, '%params'
929           hadParams = 1
930         end
931         blockHash.hasKey('%arguments'):begin
932           if ~(types[i] eq 'function') then syntaxCall += ', ' ; check if we need comma for procedure
933           syntaxCall += blockhash['%arguments']
934           blockHash.Remove, '%arguments'
935           hadParams = 1
936         end
937         else: ;do nothing
938       endcase
939
940       ; check if we have keyword syntax
941       if blockHash.hasKey('%keywords') then begin
942         ;add comma separator
943         if (hadParams OR ~(types[i] eq 'function')) then syntaxCall += ', '
944         syntaxCall += blockhash['%keywords']
945         blockHash.Remove, '%keywords'
946       endif
947
948       ;save name with function parenthesis
949       if (types[i] eq 'function') then begin
950         syntaxCall += ')'
951         commentBlocks[names[i] + '()'] = blockHash
952       endif else begin
953         commentBlocks[names[i]] = blockHash
954       endelse
```

IDL Console | Command History | Problems

Current Directory: C:\Users\Traininglead

```
IDL> .compile -v C:\Users\Traininglead\Documents\github\Tools\SE-IDLPackageCreator\src\idlpd\idlpd__define.pro
% Compiled module: IDLPD_PROCESS_MD_FILE.
% Compiled module: IDLPD_PROCESS_TXT_FILE.
% Compiled module: IDLPD_PROCESS_BLOCK.
% Compiled module: IDLPD_BLOCKS_TO_STRINGS.
% Compiled module: IDLPD_EXTRACT_BLOCKS.
% Compiled module: IDLPD::INIT.
% Compiled module: IDLPD::DOCUMENT.
% Compiled module: IDLPD__DEFINE.
IDL>
```

Writable | Smart Insert | 934 : 41

Demo!

Contest!

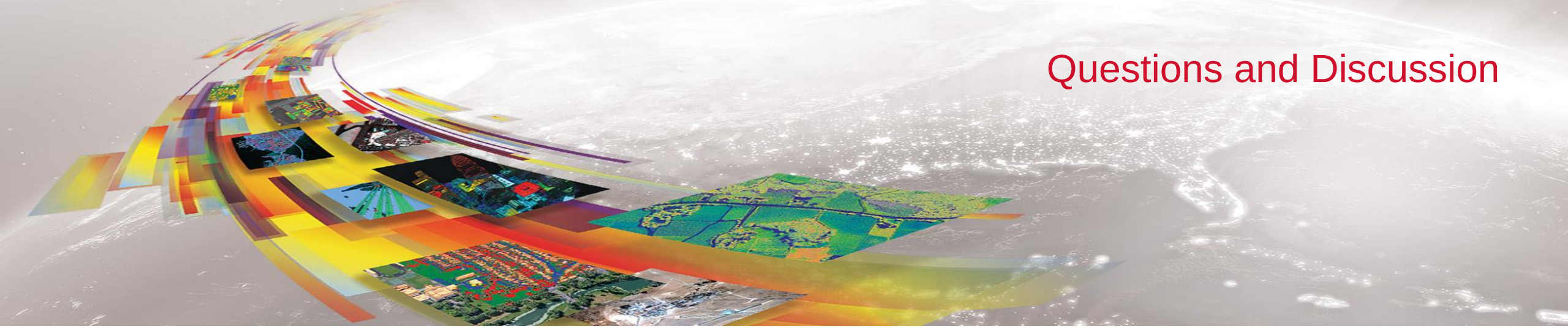
Based around open sourced IDL Packages

Two categories:

1. ENVI
2. IDL

Winners get free attendance to the ENVI Analytics Symposium and opportunity to speak at the VIP Customer Summit

Additional details coming soon on criteria and submissions



Questions and Discussion

Follow up email will be sent with whitepaper on open source technologies when the packages are placed on GitHub

https://www.harrisgeospatial.com/Company/Contact-Us

Zachary Norman

Solutions Engineer

znorman@harris.com

Harris Geospatial Solutions

HarrisGeospatial.com
geospatialinfo@harris.com
303-786-9900