
PhotoScan Python Reference

Release 0.8.5

Agisoft LLC

April 09, 2012

CONTENTS

1 Overview	3
1.1 Introduction to Python scripting in PhotoScan	3
2 Application Modules	5
2.1 PhotoScan - core functionality	5
3 Python API Change Log	27
3.1 PhotoScan version 0.8.5 build 1398	27
3.2 PhotoScan version 0.8.4 build 1289	27
3.3 PhotoScan version 0.8.3 build 1212	27
3.4 PhotoScan version 0.8.3 build 1154	28
Python Module Index	29

Copyright (c) 2012 Agisoft LLC.

OVERVIEW

1.1 Introduction to Python scripting in PhotoScan

This API is in development and will be extended in the future PhotoScan releases.

Note: Python scripting is supported only in PhotoScan Professional edition.

PhotoScan uses Python 3.2 as a scripting engine.

Python commands and scripts can be executed in PhotoScan in one of the following ways:

- From PhotoScan “Console” pane using it as standard Python console
- From the “Tools” menu using “Run script...” command

The following PhotoScan functionality can be accessed from Python scripts:

- Open/save/append PhotoScan projects, add/remove chunks, open/remove photos
- Align photos, reconstruct geometry, build texture for single or multiple chunks
- Decimate mesh
- Align and merge chunks
- Import model and texture
- Use camera calibration data
- Enable/disable chunks and photos
- Export model, sparse/dense point cloud and texture
- Load/export ground control points and assign geographic projection and coordinates
- Optimize photo alignment
- Export orthophoto and DEM
- Set camera view directions
- Export and import camera positions

The following functionality is not currently available from Python scripts:

- Export to PDF
- Resize/rotate bounding box
- Close holes in “sharp” models

APPLICATION MODULES

2.1 PhotoScan - core functionality

PhotoScan core functionality

```
import PhotoScan
```

```
doc = PhotoScan.app.document
```

```
doc.activeChunk.alignPhotos(accuracy="high", preselection="generic")
```

```
doc.activeChunk.buildModel(quality="medium", object="arbitrary", geometry="smooth", faces=50000)
```

```
doc.activeChunk.buildTexture(mapping="generic", blending="average", width=2048, height=2048)
```

```
doc.save("test2.psz")
```

PhotoScan.**alignChunks** (*chunks, reference, method='points', accuracy='high', preselection=False*)
Aligns specified set of chunks.

Parameters

- **chunks** (*list*) – List of chunks to be aligned.
- **reference** (*Chunk*) – Chunk to be used as a reference.
- **method** (*string*) – Alignment method in ['points', 'markers'].
- **accuracy** (*string*) – Alignment accuracy in ['high', 'medium', 'low'].
- **preselection** (*boolean*) – Enables image pair preselection.

Returns Success of operation.

Return type boolean

PhotoScan.**mergeChunks** (*chunks, merge_models=False, merge_markers=False*)
Merges specified set of chunks.

Parameters

- **chunks** (*list*) – List of chunks to be merged.
- **merge_models** (*boolean*) – Enables/disables merging of polygonal models.
- **merge_markers** (*boolean*) – Enables/disables merging of corresponding marker across the chunks.

Returns Merged Chunk object or None on error.

Return type `Chunk`

class `PhotoScan.Application`

Provides access to PhotoScan application.

getExistingDirectory (`[hint]`)

Prompts user for the existing folder.

Parameters `hint` (*string*) – Optional text label for the dialog.

Returns Path to the folder selected. If the input was cancelled, empty string is returned.

Return type `string`

getFloat (`[hint]`)

Prompts user for the floating point value.

Parameters `hint` (*string*) – Optional text label for the dialog.

Returns Floating point value entered by the user.

Return type `float`

getInt (`[hint]`)

Prompts user for the integer value.

Parameters `hint` (*string*) – Optional text label for the dialog.

Returns Integer value entered by the user.

Return type `int`

getOpenFileName (`[hint]`)

Prompts user for the existing file.

Parameters `hint` (*string*) – Optional text label for the dialog.

Returns Path to the file selected. If the input was cancelled, empty string is returned.

Return type `string`

getOpenFileNames (`[hint]`)

Prompts user for one or more existing files.

Parameters `hint` (*string*) – Optional text label for the dialog.

Returns List of file paths selected by the user. If the input was cancelled, empty list is returned.

Return type `list`

getSaveFileName (`[hint]`)

Prompts user for the file. The file does not have to exist.

Parameters `hint` (*string*) – Optional text label for the dialog.

Returns Path to the file selected. If the input was cancelled, empty string is returned.

Return type `string`

getString (`[hint]`)

Prompts user for the string value.

Parameters `hint` (*string*) – Optional text label for the dialog.

Returns String entered by the user.

Return type `string`

messageBox (*message*)

Displays message box to the user.

Parameters **message** (*string*) – Text message to be displayed.

quit ()

Exits the application.

update ()

Updates user interface during long operations.

document

Main application document object.

Type `Document`

version

PhotoScan version.

Type `string`

viewpoint

Viewpoint in the model view.

Type `Viewpoint`

class `PhotoScan.Calibration`

Camera calibration data

cx

Principal point X coordinate.

Type `float`

cy

Principal point Y coordinate.

Type `float`

fx

X focal length component.

Type `float`

fy

Y focal length component.

Type `float`

k1

Radial distortion coefficient K1.

Type `float`

k2

Radial distortion coefficient K2.

Type `float`

k3

Radial distortion coefficient K3.

Type `float`

p1

Tangential distortion coefficient P1.

Type float

p2

Tangential distortion coefficient P2.

Type float

skew

Skew coefficient.

Type float

class PhotoScan.**Chunk**

Chunk instance

```
import PhotoScan
```

```
doc = PhotoScan.app.document
```

```
new_chunk = PhotoScan.Chunk()
```

```
new_chunk.label = "New Chunk"
```

```
working_path = "D:/PhotoScan/IMG_000"
```

```
for i in range(1, 6):
```

```
    file_path = working_path + str(i) + ".jpg"
```

```
    new_chunk.photos.add(file_path)
```

```
new_chunk.photos.remove((len.new_chunk.photos) - 1)
```

```
new_chunk.enabled = False
```

```
doc.chunks.add(new_chunk)
```

alignPhotos (*accuracy='high', preselection='disabled', filter_mask=False*)

Performs photo alignment for the chunk.

Parameters

- **accuracy** (*string*) – Alignment accuracy in ['high', 'medium', 'low'].
- **preselection** (*string*) – Image pair preselection method in ['disabled', 'generic', 'ground control'].
- **filter_mask** (*boolean*) – Filter points by mask.

Returns Success of operation.

Return type boolean

buildModel (*quality='medium', object='arbitrary', geometry='smooth', faces=200000, filter_threshold=0.5, hole_threshold=0.1, gpu_mask=0, cpu_cores_inactive=0*)

Generates model for the chunk.

Parameters

- **quality** (*string*) – Model reconstruction quality in ['point cloud', 'lowest', 'low', 'medium', 'high', 'ultra high'].
- **object** (*string*) – Type of object to be reconstructed in ['arbitrary', 'height field'].
- **geometry** – Geometry type in ['sharp', 'smooth'].
- **faces** (*int*) – Target face count

- **filter_threshold** (*float*) – Filtering threshold.
- **hole_threshold** (*float*) – Hole filling threshold.
- **gpu_mask** (*int*) – GPU device bit mask: 1 - use device, 0 - do not use (i.e. value 5 enables device number 0 and 2).
- **cpu_cores_inactive** (*int*) – Number of CPU cores to reserve for GPU tasks during processing. It is recommended to deactivate one CPU core for each GPU in use for optimal performance.

Returns Success of operation.

Return type boolean

buildPoints (*accuracy='high', preselection='disabled', filter_mask=False*)

Rebuilds point cloud for the chunk.

Parameters

- **accuracy** (*string*) – Point cloud accuracy in ['high', 'medium', 'low'].
- **preselection** (*string*) – Image pair preselection method in ['disabled', 'generic', 'ground control'].
- **filter_mask** (*boolean*) – Filter points by mask.

Returns Success of operation.

Return type boolean

buildTexture (*mapping='generic', blending='average', width=2048, height=2048[, photo]*)

Generates texture for the chunk.

Parameters

- **mapping** (*string*) – Texture mapping mode in ['generic', 'orthophoto', 'adaptive orthophoto', 'single photo', 'keep uv'].
- **blending** (*string*) – Texture blending mode in ['mosaic', 'average', 'max', 'min'].
- **width** (*int*) – Desired texture width.
- **height** (*int*) – Desired texture height.
- **photo** (`Photo`) – Photo to be used for texturing in 'single photo' mapping mode.

Returns Success of operation.

Return type boolean

copy ()

Returns a copy of the chunk.

Returns Copy of the chunk.

Return type `Chunk`

exportCameras (*path, format='xml', projection, rotation_order='xyz'*)

Export point cloud and/or camera positions.

Parameters

- **path** (*string*) – Path to output file.
- **format** (*string*) – Export format in ['xml', 'chan', 'boujou', 'bundler', 'opk'].
- **projection** (`Matrix` or `GeoProjection`) – Sets output projection.

- **rotation_order** (*string*) – Rotation order (CHAN format only) in ['xyz', 'xzy', 'yxz', 'yzx', 'zxy', 'zyx']

Returns Success of operation.

Return type boolean

exportDem (*path*, *format='tif'* [, *projection*] [, *region*] [, *dx*] [, *dy*] [, *blockw*] [, *blockh*] [, *write_kml=False*, *write_world=False*])
Exports digital elevation model.

Parameters

- **path** (*string*) – Path to output DEM.
- **format** (*string*) – Export format in ['tif', 'asc', 'bil', 'xyz'].
- **projection** (*Matrix* or *GeoProjection*) – Sets output projection.
- **region** (*tuple of 4 floats*) – Region to be exported in the (x0, y0, x1, y1) format.
- **dx** (*float*) – Pixel size in the X dimension in projected units.
- **dy** (*float*) – Pixel size in the Y dimension in projected units.
- **blockw** (*int*) – Specifies block width of the DEM mosaic in pixels.
- **blockh** (*int*) – Specifies block height of the DEM mosaic in pixels.
- **write_kml** (*boolean*) – Enables/disables kml file generation.
- **write_world** (*boolean*) – Enables/disables world file generation.

Returns Success of operation.

Return type boolean

exportOrthophoto (*path*, *format='tif'*, *blending='mosaic'* [, *projection*] [, *region*] [, *dx*] [, *dy*] [, *blockw*] [, *blockh*] [, *write_kml=False*, *write_world=False*])
Exports orthophoto for the chunk.

Parameters

- **path** (*string*) – Path to output orthophoto.
- **format** (*string*) – Export format in ['tif', 'jpg', 'png', 'kml'].
- **blending** (*string*) – Orthophoto blending mode in ['mosaic', 'average', 'max', 'min'].
- **projection** (*Matrix* or *GeoProjection*) – Sets output projection.
- **region** (*tuple of 4 floats*) – Region to be exported in the (x0, y0, x1, y1) format.
- **dx** (*float*) – Pixel size in the X dimension in projected units.
- **dy** (*float*) – Pixel size in the Y dimension in projected units.
- **blockw** (*int*) – Specifies block width of the orthophoto mosaic in pixels.
- **blockh** (*int*) – Specifies block height of the orthophoto mosaic in pixels.
- **write_kml** (*boolean*) – Enables/disables kml file generation.
- **write_world** (*boolean*) – Enables/disables world file generation.

Returns Success of operation.

Return type boolean

exportPoints (*path*, *format*='obj', *quality*='point_cloud'[, *projection*])
Exports point cloud.

Parameters

- **path** (*string*) – Path to output file.
- **format** (*string*) – Export format in ['obj', 'ply', 'xyz', 'las'].
- **quality** (*string*) – Point cloud quality in ['point cloud', 'lowest', 'low', 'medium', 'high', 'ultra high'].
- **projection** (*Matrix* or *GeoProjection*) – Sets output projection.

Returns Success of operation.

Return type boolean

importCameras (*path*, *format*='xml')
Imports camera positions.

Parameters

- **path** (*string*) – Path to the file.
- **format** (*string*) – File format in ['xml', 'bundler'].

Returns Success of operation.

Return type boolean

optimize (*fit_aspect*=True, *fit_skew*=True, *fit_p1p2*=True)
Performs optimization of point cloud / camera parameters.

Parameters

- **fit_aspect** (*boolean*) – Specifies if aspect calibration coefficients should be optimized.
- **fit_skew** (*float*) – Specifies if skew calibration coefficients should be optimized.
- **fit_p1p2** (*float*) – Specifies if tangential distortion coefficients should be optimized.

Returns Success of operation.

Return type boolean

aligned_count
Number of aligned photos in the chunk.

Type int

enabled
Enables/disables the chunk.

Type boolean

fix_calibration
Sets fix calibration flag.

Type boolean

ground_control
Ground control data for the chunk.

Type `GroundControl`

label
Chunk label.

Type string

markers

List of markers in the chunk.

Type `Markers`

model

Generated model.

Type `Model`

photos

List of photos in the chunk.

Type `Photos`

point_cloud

Generated sparse point cloud.

Type `PointCloud`

projection

Geographic coordinate system used as a world coordinate system.

Type `GeoProjection`

selected

Selects/deselects the chunk.

Type boolean

transform

4x4 matrix specifying chunk location in the world coordinate system.

Type `Matrix`

class `PhotoScan.Chunks` (*doc*)

Collection of chunks in the document

add (*chunk=None*)

Adds new chunk to the document.

Parameters **chunk** (`Chunk`) – Optional argument specifying the chunk to be added. An empty chunk is added if unspecified.

Returns Index of the added chunk.

Return type int

index (*chunk*)

Returns index of the specified chunk.

Parameters **chunk** (`Chunk`) – Chunk to be looked for.

Returns Index of the chunk in the document.

Return type int

remove (*chunk*)

Removes specified chunk from the document.

Parameters **chunk** (`Chunk` or int) – Chunk object to be removed or index in the list of chunks.

Returns Success of operation.

Return type boolean

class PhotoScan.Document

Represents PhotoScan document

```
import PhotoScan

main_doc = PhotoScan.app.document

new_doc = PhotoScan.Document ()
new_doc.open ("D:/PhotoScan/test2.psz")

for i in range (1,4):
    new_doc.chunks.add()

main_doc.append(new_doc)

main_doc.active = len(main_doc.chunks) - 4
```

append (*document*)

Appends the specified Document object to the current document.

Parameters *document* (*Document*) – document object to be appended.**Returns** Success of operation.**Return type** boolean**clear** ()

Clears the contents of the Document object.

Returns Success of operation.**Return type** boolean**open** (*path*)

Loads document from the specified file.

Parameters *path* (*string*) – Path to the file.**Returns** Success of operation.**Return type** boolean**save** ([*path*])

Saves document to the specified file.

Parameters *path* (*string*) – path to the file (optional).**Returns** Success of operation.**Return type** boolean**active**

Index of the active chunk.

Type int**activeChunk**

Active Chunk.

Type *Chunk***chunks**

List of chunks in the document.

Type *Chunks*

path

Path to the document file.

Type string

class PhotoScan.**GeoProjection**

Provides access to geographic projections

```
import PhotoScan
```

```
chunk = PhotoScan.Chunk()
```

```
proj = PhotoScan.GeoProjection()  
proj.init("EPSG::32641")
```

```
gc = chunk.ground_control  
gc.projection = proj  
gc.load("gcp.txt", "csv")
```

```
gc.apply()
```

init (*crs*)

Initialize projection based on specified WKT definition or authority identifier.

Parameters *crs* (*string*) – WKT definition of coordinate system or authority identifier.

Returns Success of operation.

Return type boolean

localframe (*point*)

Returns 4x4 matrix with a local coordinates at the given point.

Parameters *point* (*Vector*) – Coordinates of the origin in the geocentric coordinates.

Returns Transformation from geocentric coordinates to local coordinates.

Return type *Matrix*

project (*point*)

Projects point from geocentric coordinates to projected geographic coordinate system.

Parameters *point* (*Vector*) – 3D point in geocentric coordinates.

Returns 3D point in projected coordinates.

Return type *Vector*

unproject (*point*)

Unprojects point from projected coordinates to geocentric coordinates.

Parameters *point* (*Vector*) – 3D point in projected coordinate system.

Returns 3D point in geocentric coordinates.

Return type *Vector*

authority

Authority identifier of the coordinate system.

Type string

wkt

WKT string identifier of the coordinate system.

Type string

class PhotoScan.**GroundControl** (*chunk*)

Provides access to the ground control data for the chunk

apply ()

Updates chunk transformation based on the ground control data.

Returns Success of operation.

Return type boolean

load (*path, format*)

Imports ground control data from the specified file.

Parameters

- **path** (*string*) – Path to the file with ground control data.
- **format** (*string*) – Format of the file in ['xml', 'tel', 'csv', 'mavinci']

Returns Success of operation.

Return type boolean

loadExif ()

Returns a copy of the model.

Returns Copy of the model.

Return type Model

save (*path, format*)

Exports ground control data to the specified file.

Parameters

- **path** (*string*) – Path to the output file.
- **format** (*string*) – Export format in ['xml', 'tel', 'csv'].

Returns Success of operation.

Return type boolean

accuracy_cameras

Expected accuracy of camera coordinates in meters.

Type float

accuracy_markers

Expected accuracy of marker coordinates in meters.

Type float

accuracy_projections

Expected accuracy of marker projections in pixels.

Type float

locations

Ground control coordinates.

Type GroundControlLocations

projection

Ground control projection.

Type GeoProjection

class PhotoScan.**GroundControlLocation** (*chunk, photo*)
Provides access to the ground control coordinates for the given photo

coord
Keypoint coordinates.
Type tuple of 3 float

enabled
Enabled flag.
Type boolean

transform
Transformation matrix.
Type class‘Matrix‘

class PhotoScan.**GroundControlLocations** (*chunk*)
Collection of ground control locations in the chunk

add (*item*)
Adds a ground control record for a given photo or marker.
Parameters *item* (*Photo* or *Marker*) – Photo or Marker instance.

items ()
List of items.

keys ()
List of item keys.

values ()
List of item values.

class PhotoScan.**Image** (*width, height, cn*)
1 or 3-channel bitmap

copy ()
Makes a copy of the image.
Returns copy of the image
Return type *Image*

load (*path*)
Loads image from the file.
Parameters *path* (*string*) – path to the image file
Returns success of operation
Return type boolean

save (*path*)
Saves image to the file.
Parameters *path* (*string*) – path to the image file
Returns success of operation
Return type boolean

undistort (*calib, center_principal_point = True, square_pixels = True*)
Undistorts image using provided calibration.
Parameters

- **calib** (*Calibration*) – lens calibration
- **center_principal_point** (*boolean*) – moves principal point to the image center
- **square_pixels** (*boolean*) – create image with square pixels

Returns undistorted image

Return type *Image*

cn

Number of color channels (1 or 3).

Type *int*

height

Image height.

Type *int*

width

Image width.

Type *int*

class *PhotoScan*.**Marker**

Marker instance

copy ()

Returns a copy of the marker.

Returns Copy of the marker.

Return type *Marker*

label

Marker label.

Type *string*

projections

List of marker projections.

Type *MarkerProjections*

selected

Selects/deselects the marker.

Type *boolean*

class *PhotoScan*.**MarkerProjections** (*marker*)

Collection of projections specified for the marker

items ()

List of items.

keys ()

List of item keys.

values ()

List of item values.

class *PhotoScan*.**Markers** (*chunk*)

Collection of markers in the chunk

add (*marker=None*)

Adds new marker to the chunk.

Parameters `marker` (`Marker`) – Optional argument specifying the marker to be added. An empty marker is added if unspecified.

Returns Index of the added marker.

Return type `int`

index (`marker`)

Returns index of the specified marker.

Parameters `marker` (`Marker`) – Marker to be looked for.

Returns Index of the marker or -1 if marker is not found in the chunk.

Return type `int`

remove (`marker`)

Removes specified marker from the chunk.

Parameters `marker` (`Marker` or `int`) – Marker object to be removed or index in the list of markers.

Returns Success of operation.

Return type `boolean`

class `PhotoScan.Mask` (`photo`)

Mask for the photo

fromAlpha ()

Imports mask from alpha channel.

Returns Success of operation.

Return type `boolean`

fromImage (`image`)

Imports mask from a bitmap representation.

Parameters `image` (`Image`) – Bitmap representation of the mask.

toImage ()

Returns a bitmap representation of the vector mask.

Returns Generated bitmap representation.

Return type `Image`

contours

Collection of contours.

Type `MaskContours`

class `PhotoScan.MaskContour` (`photo`, `index`)

Contour of the mask

class `PhotoScan.MaskContours` (`photo`)

Collection of mask contours

class `PhotoScan.Matrix`

m-by-n matrix

```
import PhotoScan
```

```
m1 = PhotoScan.Matrix.diag( (1,2,3,4) )
```

```
m3 = PhotoScan.Matrix( [[1,2,3,4], [1,2,3,4], [1,2,3,4], [1,2,3,4]] )
```

```

m2 = m1.inv()
m3 = m1 * m2

x = m3.det()

if x == 1:
    PhotoScan.app.messageBox("Diagonal matrix dimensions: " + str(m3.size))

```

classmethod `diag` (*vector*)

Create a diagonal matrix.

Parameters **vector** (*Vector* or list of floats) – The vector of diagonal entries.

Returns A diagonal matrix.

Return type *Matrix*

classmethod `translation` (*vector*)

Create a translation matrix.

Parameters **vector** (*Vector*) – The translation vector.

Returns A matrix representing translation.

Return type *Matrix*

`copy` ()

Returns a copy of this matrix.

Returns an instance of itself

Return type *Matrix*

`det` ()

Return the determinant of a matrix.

Returns Return a the determinant of a matrix.

Return type float

`inv` ()

Return an inverted copy of the matrix.

Returns the inverted matrix.

Return type *Matrix*

`t` ()

Return a new, transposed matrix.

Returns a transposed matrix

Return type *Matrix*

`zero` ()

Set all matrix elements to zero.

`size`

Matrix dimensions.

Type tuple

class `PhotoScan.MeshFace` (*model, index*)

Triangular face of the model

hidden

Face visibility flag.

Type boolean

selected

Face selection flag.

Type boolean

tex_vertices

Texture vertex indices.

Type tuple of 3 int

vertices

Vertex indices.

Type tuple of 3 int

class PhotoScan.**MeshFaces** (*model*)

Collection of model faces

class PhotoScan.**MeshTexVertex** (*model, index*)

Texture vertex of the model

coord

Vertex coordinates.

Type tuple of 2 float

class PhotoScan.**MeshTexVertices** (*model*)

Collection of model texture vertices

class PhotoScan.**MeshVertex** (*model, index*)

Vertex of the model

color

Vertex color.

Type tuple of 3 int

coord

Vertex coordinates.

Type *Vector*

class PhotoScan.**MeshVertices** (*model*)

Collection of model vertices

class PhotoScan.**Model**

Triangular mesh model instance

copy ()

Returns a copy of the model.

Returns Copy of the model.

Return type *Model*

cropSelection ()

Crops selected faces and free vertices from the mesh.

decimate (*face_count*)

Decimates the model to the specified face count.

Parameters `face_count` (*int*) – Target face count.

Returns Success of operation.

Return type boolean

load (*path, format*)

Imports model from file.

Parameters

- **path** (*string*) – Path to model.
- **format** (*string*) – Model format in ['obj', 'ply'].

Returns Success of operation.

Return type boolean

loadTexture (*path*)

Loads texture from the specified file.

Parameters **path** (*string*) – Path to the image file.

Returns Success of operation.

Return type boolean

removeSelection ()

Remove selected faces and free vertices from the mesh.

save (*path, format, texture_format='jpg', export_texture=True, export_normals=True, export_cameras=True, split_chunks=True*)

Exports generated model for the chunk.

Parameters

- **path** (*string*) – Path to output model.
- **format** (*string*) – Export format in ['3ds', 'collada', 'dfx', 'obj', 'pdf', 'ply', 'u3d', 'vrmf'].
- **texture_format** (*string*) – Texture format in ['jpg', 'png'].
- **export_texture** (*boolean*) – Enables/disables texture export.
- **export_normals** (*boolean*) – Enables/disables export of vertex normals.
- **export_cameras** (*boolean*) – Enables/disables camera export.
- **split_chunks** (*boolean*) – Split model in chunks ('collada' format only)

Returns Success of operation.

Return type boolean

saveTexture (*path*)

Saves texture to the specified file.

Parameters **path** (*string*) – Path to the image file.

Returns Success of operation.

Return type boolean

faces

Collection of mesh faces.

Type `MeshFaces`

tex_vertices

Collection of mesh texture vertices.

Type `MeshTexVertices`

vertices

Collection of mesh vertices.

Type `MeshVertices`

class `PhotoScan.Photo`

Photo instance

```
import PhotoScan
```

```
doc = PhotoScan.app.document
```

```
p = PhotoScan.Photo()
```

```
p.open = "D:/PhotoScan/IMG_0001.jpg"
```

```
x = p.width * p.height
```

```
num = doc.chunks.add()
```

```
doc.active = num
```

```
doc.activeChunk.photos.add(p)
```

```
PhotoScan.app.messageBox("Opened " + str(round(x / 1000000, 2)) + " MPix photo.")
```

copy()

Returns a copy of the photo.

Returns Copy of the photo.

Return type `Photo`

open (*path* [, *index*])

Loads specified image file.

Parameters

- **path** (*string*) – Path to the image file to be loaded.
- **index** (*int*) – Optional image index in case of MPO files.

Returns Success of operation.

Return type `boolean`

project (*point*)

Returns coordinates of the point projection on the photo.

Parameters **point** (`Vector`) – Coordinates of the point to be projected.

Returns 2D point coordinates.

Return type tuple of 2 floats

calibration

Refined calibration of the photo.

Type `Calibration`

center

Camera station coordinates for the photo in the chunk coordinate system.

Type `Vector`

enabled
Enables/disables the photo.

Type `boolean`

height
Image height.

Type `int`

image
Image object with loaded pixel data.

Type `Image`

mask
Mask for the photo.

Type `Mask`

path
Path to the image file.

Type `string`

selected
Selects/deselects the photo.

Type `boolean`

ttransform
4x4 matrix describing photo location in the chunk coordinate system.

Type `Matrix`

user_calib
Custom calibration used as initial calibration during photo alignment.

Type `Calibration`

width
Image width.

Type `int`

class `PhotoScan.Photos` (*chunk*)
Collection of photos in the chunk

add (*photo*)
Adds a photo to the chunk.

Parameters **photo** (`Photo`, string or list of strings) – Photo object, path to the image file or list of paths to the photos.

Returns Success of operation.

Return type `boolean`

index (*photo*)
Returns index of the specified photo.

Parameters **photo** (`Photo`) – Photo to be looked for.

Returns Index of the photo.

Return type int

remove (*photo*)

Removes specified photo from the chunk.

Parameters **photo** (`Photo` or int) – Photo object to be removed or index in the list of photos.

Returns Success of operation.

Return type boolean

class `PhotoScan.PointCloud`

Sparse point cloud instance

copy ()

Returns a copy of the point cloud.

Returns Copy of the point cloud.

Return type `PointCloud`

export (*path*, *format*='obj'[, *projection*])

Export point cloud.

Parameters

- **path** (*string*) – Path to output file.
- **format** (*string*) – Export format in ['obj', 'ply'].
- **projection** (`Matrix` or `GeoProjection`) – Sets output projection.

Returns Success of operation.

Return type boolean

points

List of points.

Type `PointCloudPoints`

projections

Point projections for each photo.

Type `PointCloudPhotos`

class `PhotoScan.PointCloudPhotos` (*point_cloud*)

Collection of `PointCloudProjections` objects indexed by corresponding photos

class `PhotoScan.PointCloudPoint` (*point_cloud*, *index*)

3D point in the point cloud

color

Point color.

Type tuple of 3 int

coord

Point coordinates.

Type tuple of 3 float

selected

Point selection flag.

Type boolean

valid

Point valid flag.

Type boolean

class PhotoScan.**PointCloudPoints** (*point_cloud*)
Collection of 3D points in the point cloud

class PhotoScan.**PointCloudProjection** (*point_cloud, photo, index*)
Projection of the 3D point on the photo

coord

Projection coordinates.

Type tuple of 2 float

index

Point index.

Type int

class PhotoScan.**PointCloudProjections** (*point_cloud, photo*)
Collection of **PointCloudProjection** for the photo

class PhotoScan.**Vector**
n-component vector

```
import PhotoScan
```

```
vect = PhotoScan.Vector( (1, 2, 3) )
vect2 = vect.copy()
```

```
vect2.size = 4
vect2.w = 5
vect2 *= -1.5
```

```
vect.size = 4
vect.normalize()
```

```
PhotoScan.app.messageBox("Scalar product is " + str(vect2 * vect))
```

copy()

Returns a copy of the vector.

Returns A copy of the vector.

Return type **Vector**

normalize()

Normalizes vector to the unit length.

normalized()

Return a new, normalized vector.

Returns a normalized copy of the vector

Return type **Vector**

zero()

Sets all elements to zero.

size

Vector dimensions.

Type int

w Vector W component.

Type float

x Vector X component.

Type float

y Vector Y component.

Type float

z Vector Z component.

Type float

class PhotoScan.**Viewpoint** (*app*)
Represents viewpoint in the model view

coo Center of orbit.

Type *Vector*

fov Camera vertical field of view in degrees.

Type float

mag Camera magnification defined by distance to the center of rotation.

Type float

rot Camera rotation matrix.

Type *Matrix*

PYTHON API CHANGE LOG

3.1 PhotoScan version 0.8.5 build 1398

- Added `Chunk.fix_calibration` property
- Removed “`fix_calibration`” parameter from `Chunk.alignPhotos()` method
- Added `Chunk.exportCameras()` method
- Added `Chunk.exportPoints()` method for dense/sparse point cloud export
- Moved `GroundControl.optimize()` method to `Chunk.optimize()`
- Added `accuracy_cameras`, `accuracy_markers` and `accuracy_projections` properties to the `GroundControl` class
- Added `Image.undistort()` method
- Added `PointCloudPoint.selected` and `PointCloudPoint.valid` properties
- Removed `GeoProjection.epsg` property
- Added `GeoProjection.authority` property
- Added `GeoProjection.init()` method

3.2 PhotoScan version 0.8.4 build 1289

- Added `GroundControl.optimize()` method
- Command line scripting support removed

3.3 PhotoScan version 0.8.3 build 1212

- Revised class: `Chunk`
- Added classes: `Model`, `PointCloud`, `Image`
- `alignPhotos()`, `buildModel()` and `buildTexture()` are now methods of `Chunk` class
- Added export support for point cloud, orthophoto and DEM
- Added `GroundControl` class

3.4 PhotoScan version 0.8.3 build 1154

Initial version of PhotoScan Python API

PYTHON MODULE INDEX

p

PhotoScan, 5