PhotoScan Python Reference

Release 1.0.0

Agisoft LLC

December 24, 2013

CONTENTS

CHAPTER

ONE

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 3.3 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 funtionality can be accessed from Python scripts:

- Open/save/create PhotoScan projects
- Add/remove chunks, cameras, markers
- Add/modify camera calibrations, ground control data, assign geographic projections and coordinates
- Perform processing steps (align photos, build dense cloud, build mesh, texture, decimate model, etc...)
- Export processing results (models, textures, orthophotos, DEMs)
- Access data of generated models, point clouds, images

CHAPTER

TWO

APPLICATION MODULES

2.1 PhotoScan - core functionality

PhotoScan core functionality

```
import PhotoScan

doc = PhotoScan.app.document

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

doc.activeChunk.alignPhotos()

doc.activeChunk.buildDepth(quality="medium")

doc.activeChunk.buildModel(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'*, *fix_scale=False*, *accuracy='high'*, *preselection=False*, *filter_mask=False*, *point_limit=40000*)

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', 'cameras'].
- **fix_scale** (*boolean*) – Fixes chunk scale during alignment.
- **accuracy** (*string*) – Alignment accuracy in ['high', 'medium', 'low'].
- **preselection** (*boolean*) – Enables image pair preselection.
- **filter_mask** (*boolean*) – Filter points by mask.
- **point_limit** (*int*) – Maximum number of points for each photo.

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.

addMenuItem(*label*, *func*[, *shortcut*])

Creates a new menu entry.

Parameters

- **label** (*string*) – Menu item label.
- **func** (*function*) – Function to be called.
- **shortcut** (*string*) – Keyboard shortcut.

enumOpenCLDevices()

Returns a list of installed OpenCL devices.

Returns A list of devices.

Return type list

getCoordinateSystem([*label*][, *value*])

Prompts user for coordinate system.

Parameters

- **label** (*string*) – Optional text label for the dialog.
- **value** (CoordinateSystem) – Default value.

Returns Selected coordinate system. If the dialog was cancelled, None is returned.

Return type CoordinateSystem

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(*label='‘*, *value=0*)

Prompts user for the floating point value.

Parameters

- **label** (*string*) – Optional text label for the dialog.
- **value** (*float*) – Default value.

Returns Floating point value entered by the user.

Return type float

`getInt` (*label='‘*, *value=0*)

Prompts user for the integer value.

Parameters

- **label** (*string*) – Optional text label for the dialog.
- **value** (*int*) – Default value.

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` (*label='‘*, *value='‘*)

Prompts user for the string value.

Parameters

- **label** (*string*) – Optional text label for the dialog.
- **value** (*string*) – Default value.

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

`load`(*path*)

Loads calibration from file.

Parameters **path** (*string*) – path to calibration file

Returns success of operation

Return type boolean

`project`(*point*)

Returns projected pixel coordinates of the point.

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

Returns 2D projected point coordinates.

Return type `Vector`

`save`(*path*)

Saves calibration to file.

Parameters **path** (*string*) – path to calibration file

Returns success of operation

Return type boolean

`unproject`(*point*)

Returns direction corresponding to the image point.

Parameters **point** (`Vector`) – Pixel coordinates of the point.

Returns 3D vector in the camera coordinate system.

Return type `Vector`

`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

height
Image height.

Type int

k1
Radial distortion coefficient K1.

Type float

k2
Radial distortion coefficient K2.

Type float

k3
Radial distortion coefficient K3.

Type float

k4
Radial distortion coefficient K4.

Type float

p1
Tangential distortion coefficient P1.

Type float

p2
Tangential distortion coefficiant P2.

Type float

skew
Skew coefficient.

Type float

width
Image width.

Type int

class PhotoScan.**Camera**
Camera instance

```
import PhotoScan

doc = PhotoScan.app.document

p = PhotoScan.Camera()

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

x = p.width * p.height

num = doc.chunks.add()
doc.active = num
doc.activeChunk.cameras.add(p)

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

alpha()

Returns alpha channel data.

Returns Alpha channel data.

Return type `Image`

append(*path*[, *layer*])

Appends a new frame to the camera.

Parameters

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

Returns Success of operation.

Return type boolean

copy()

Returns a copy of the photo.

Returns Copy of the photo.

Return type `Photo`

image()

Returns image data.

Returns Image data.

Return type `Image`

insert(*index*, *path*[, *layer*])

Inserts a new frame to the camera.

Parameters

- **index** (*int*) – Position in the list of frames where the new frame should be inserted.
- **path** (*string*) – Path to the image file to be loaded.
- **layer** (*int*) – Optional layer index in case of multipage files.

Returns Success of operation.

Return type boolean

mask()

Returns mask data.

Returns Mask data.

Return type `Image`

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

Loads specified image file.

Parameters

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

setMask (*mask*)

Initializes mask from image data.

Parameters **mask** (`Image`) – Mask image.

Returns Success of operation.

Return type boolean

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

frames

Camera frames.

Type `Photos`

height

Image height.

Type int

label

Camera label.

Type string

layer

Layer index in the image file.

Type int

meta

Camera meta data.

Type `MetaData`

path

Path to the image file.

Type string

selected

Selects/deselects the photo.

Type boolean

sensor

Camera sensor.

Type Sensor

transform

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.**Cameras**(*chunk*)

Collection of cameras in the chunk

add(*camera*)

Adds camera to the chunk.

Parameters **camera** (Camera, list of Camera, string or list of strings) – Camera object, list of Camera objects, path to the image file or list of paths to image files.

Returns Success of operation.

Return type boolean

index(*camera*)

Returns index of the specified camera.

Parameters **camera** (Camera) – Camera to be looked for.

Returns Index of the camera.

Return type int

remove(*camera*)

Removes specified camera from the chunk.

Parameters **camera** (Camera, list of Camera or int) – Camera object, list of Camera objects or index in the list of cameras.

Returns Success of operation.

Return type boolean

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.cameras.add(file_path)

new_chunk.cameras.remove((len.new_chunk.cameras) - 1)

new_chunk.enabled = False

doc.chunks.add(new_chunk)
```

alignPhotos([*cameras*][, *min_image*])

Performs photo alignment for the chunk.

Parameters

- **cameras** (list of `Camera`) – A list of cameras to be aligned to the existing cameras.
- **min_image** (*int*) – Minimum number of point projections.

Returns Success of operation.

Return type boolean

buildDenseCloud(*quality='medium'*, *filter='aggressive'*, *gpu_mask=0*, *cpu_cores_inactive=0*[, *frames*][, *cameras*])

Generates depth maps for the chunk.

Parameters

- **quality** (*string*) – Depth map quality in ['lowest', 'low', 'medium', 'high', 'ultra'].
- **filter** (*string*) – Depth map filtering level in ['mild', 'moderate', 'aggressive'].
- **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.
- **frames** (*list of int*) – A list of frames to be processed.
- **cameras** (list of `Camera`) – A list of cameras to be processed.

Returns Success of operation.

Return type boolean

buildModel(*surface='arbitrary'*, *source='sparse'*, *interpolation='enabled'*, *faces='medium'*[, *frames*])

Generates model for the chunk.

Parameters

- **surface** (*string*) – Type of object to be reconstructed in ['arbitrary', 'height field'].
- **source** (*string*) – Source data in ['sparse', 'dense'].
- **interpolation** (*string*) – Interpolation mode in ['disabled', 'enabled', 'extrapolated'].
- **faces** (*string or int*) – Target face count in ['low', 'medium', 'high'] or exact number.
- **frames** (*list of int*) – A list of frames to be processed.

Returns Success of operation.

Return type boolean

buildPoints(*error=10*[, *min_image*])
Rebuilds point cloud for the chunk.

Parameters

- **error** (*float*) – Reprojection error threshold.
- **min_image** (*int*) – Minimum number of point projections.

Returns Success of operation.

Return type boolean

buildTexture(*mapping='generic'*, *blending='average'*, *color_correction=False*, *size=2048*, *count=1*[, *camera*][, *frames*])
Generates texture for the chunk.

Parameters

- **mapping** (*string*) – Texture mapping mode in ['generic', 'orthophoto', 'adaptive', 'spherical', 'camera', 'current'].
- **blending** (*string*) – Texture blending mode in ['mosaic', 'average', 'max', 'min'].
- **color_correction** (*boolean*) – Enables color correction.
- **size** (*int*) – Texture size.
- **count** (*int*) – Texture count.
- **camera** (`Camera`) – Camera to be used for texturing in 'camera' mapping mode.
- **frames** (*list of int*) – A list of frames to be processed.

Returns Success of operation.

Return type boolean

copy()
Returns a copy of the chunk.

Returns Copy of the chunk.

Return type `Chunk`

decimateModel(*face_count*[, *frames*])
Decimates the model to the specified face count.

Parameters

- **face_count** (*int*) – Target face count.
- **frames** (*list of int*) – A list of frames to be processed.

Returns Success of operation.

Return type boolean

detectMarkers(*type='12bit'*, *tolerance=50*[, *frames*])
Create markers from coded targets.

Parameters

- **type** (*string*) – Coded targets type in ['12bit', '16bit'].
- **tolerance** (*int*) – Detector tolerance (0 - 100).
- **frames** (*list of int*) – A list of frames to be processed.

Returns Success of operation.

Return type boolean

`estimateImageQuality`([*cameras*])

Estimates image quality.

Parameters **cameras** (list of `Camera`) – Optional list of cameras to be processed.

Returns Success of operation.

Return type boolean

`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', 'patb', 'bingo', 'aerosys', 'inpho'].
- **projection** (`Matrix` or `CoordinateSystem`) – 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 `CoordinateSystem`) – 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

`exportModel`(*path*, *binary=True*, *precision=6*, *texture_format='jpg'*, *texture=True*, *normals=True*, *colors=True*, *cameras=True*[, *comment*][, *format*][, *projection*][, *shift*][, *frame*])

Exports generated model for the chunk.

Parameters

- **path** (*string*) – Path to output model.
- **binary** (*boolean*) – Enables/disables binary encoding (if supported by format).

- **precision** (*int*) – Number of digits afer the decimal point (for text formats).
- **texture_format** (*string*) – Texture format in ['jpg', 'png', 'tif', 'exr', 'bmp'].
- **texture** (*boolean*) – Enables/disables texture export.
- **normals** (*boolean*) – Enables/disables export of vertex normals.
- **colors** (*boolean*) – Enables/disables export of vertex colors.
- **cameras** (*boolean*) – Enables/disables camera export.
- **comment** (*string*) – Optional comment (if supported by selected format).
- **format** (*string*) – Export format in ['3ds', 'obj', 'ply', 'vrml', 'collada', 'dxf', 'fbx', 'pdf', 'u3d', 'kmz'].
- **projection** (`CoordinateSystem`) – Output coordinate system.
- **shift** (*3-element vector*) – Optional shift to be applied to vertex coordinates.
- **frame** (*int*) – Frame number to be exported (current frame if not specified).

Returns Success of operation.

Return type boolean

exportOrthophoto (*path*, *format='tif'*, *blending='mosaic'*, *color_correction=False*[, *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', 'kmz'].
- **blending** (*string*) – Orthophoto blending mode in ['mosaic', 'average', 'max', 'min'].
- **color_correction** (*boolean*) – Enables color correction.
- **projection** (`Matrix` or `CoordinateSystem`) – 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*, *dense=False*, *binary=True*, *precision=6*, *normals=True*, *colors=True*[, *comment*][, *format*][, *projection*][, *shift*][, *frame*])

Exports point cloud.

Parameters

- **path** (*string*) – Path to output file.

- **dense** (*boolean*) – Selects between dense point cloud and sparse point cloud.
- **binary** (*boolean*) – Enables/disables binary encoding for selected format (if applicable).
- **precision** (*int*) – Number of digits afer the decimal point (for text formats).
- **normals** (*boolean*) – Enables/disables export of point normals.
- **colors** (*boolean*) – Enables/disables export of point colors.
- **comment** (*string*) – Optional comment (if supported by selected format).
- **format** (*string*) – Export format in ['obj', 'ply', 'xyz', 'las', 'u3d', 'pdf'].
- **projection** (`CoordinateSystem`) – Output coordinate system.
- **shift** (*3-element vector*) – Optional shift to be applied to vertex coordinates.
- **frame** (*int*) – Frame number to be exported (current frame if not specified).

Returns Success of operation.

Return type boolean

`exportReport`(*path*)

Exports processing report in PDF format.

Parameters **path** (*string*) – Path to output report.

Returns Success of operation.

Return type boolean

`extractFrames`(*frames*)

Returns a new chunk containing a set of frames extracted from multiframe chunk.

Parameters **frames** (*list of int*) – A list of frames to be extracted.

Returns New chunk with specified frames.

Return type `Chunk`

`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

`importModel`(*path*[, *format*][, *projection*][, *shift*][, *frame*])

Imports model from file.

Parameters

- **path** (*string*) – Path to model.
- **format** (*string*) – Model format in ['obj', 'ply'].
- **projection** (`CoordinateSystem`) – Model coordinate system.
- **shift** (*3-element vector*) – Optional shift to be applied to vertex coordinates.
- **frame** (*int*) – Frame number to be imported (current frame if not specified).

Returns Success of operation.

Return type boolean

matchPhotos (*accuracy='high'*, *preselection='disabled'*, *filter_mask=False*, *point_limit=40000*)
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.
- **point_limit** (*int*) – Maximum number of points for each photo.

Returns Success of operation.

Return type boolean

optimizePhotos (*fit_f=True*, *fit_cxcy=True*, *fit_aspect=True*, *fit_skew=True*, *fit_k1k2k3=True*, *fit_p1p2=True*, *fit_k4=False*)
Performs optimization of point cloud / camera parameters.

Parameters

- **fit_f** (*boolean*) – Enables optimization of focal length coefficient.
- **fit_cxcy** (*boolean*) – Enables optimization of principal point coordinates.
- **fit_aspect** (*boolean*) – Enabled optimization of aspect ratio.
- **fit_skew** (*boolean*) – Enables optimization of skew coefficient.
- **fit_k1k2k3** (*boolean*) – Enables optimization of k1, k2 and k3 radial distortion coefficients.
- **fit_p1p2** (*boolean*) – Enables optimization of p1 and p2 tangential distortion coefficients.
- **fit_k4** (*boolean*) – Enables optimization of k4 radial distortion coefficient.

Returns Success of operation.

Return type boolean

refineMatches (*filter_mask=False*, *point_limit=40000*)
Performs precise matching.

Parameters

- **filter_mask** (*boolean*) – Filter points by mask.
- **point_limit** (*int*) – Maximum number of points for each photo.

Returns Success of operation.

Return type boolean

removeFrames (*frames*)
Removes a set of frames from multiframe chunk.

Parameters **frames** (*list of int*) – A list of frames to be removed.

Returns Success of operation.

Return type boolean

resetDepth([*frames*])

Removes depth maps for the chunk.

Parameters **frames** (*list of int*) – A list of frames to be processed.

Returns Success of operation.

Return type boolean

resetRegion()

Resets reconstruction volume selector to default position.

smoothModel(*passes = 3*[, *frames*])

Smooths mesh using Laplacian smoothing algorithm.

Parameters

- **passes** (*int*) – Number of smoothing passes to perform.
- **frames** (*list of int*) – A list of frames to be processed.

Returns Success of operation.

Return type boolean

trackMarkers([*start*][, *end*])

Tracks marker projections through the frame sequence.

Parameters

- **start** (*int*) – Starting frame index.
- **end** (*int*) – Ending frame index.

Returns Success of operation.

Return type boolean

aligned_count

Number of aligned photos in the chunk.

Type int

calibration_mode

Calibration mode in [’fixed’, ‘groups’, ‘separate’].

Type string

cameras

List of cameras in the chunk.

Type Cameras

crs

Geographic coordinate system used as a world coordinate system.

Type CoordinateSystem

dense_cloud

Generated dense point cloud for the current frame.

Type DenseCloud

dense_clouds

Generated dense point clouds for each frame.

Type DenseClouds

enabled
Enables/disables the chunk.

Type boolean

fix_calibration
Sets fix calibration flag (deprecated, use calibration_mode instead).

Type boolean

frame
Current frame index.

Type int

frame_count
Number of frames in the chunk.

Type int

ground_control
Ground control data for the chunk.

Type GroundControl

label
Chunk label.

Type string

markers
List of markers in the chunk.

Type Markers

meta
Chunk meta data.

Type MetaData

model
Generated model for the current frame.

Type Model

models
Generated models for each frame.

Type Models

photos
List of cameras in the chunk (deprecated, use cameras instead).

Type Cameras

point_cloud
Generated sparse point cloud.

Type PointCloud

projection
Geographic coordinate system used as a world coordinate system.

Type CoordinateSystem

region
Reconstruction volume selection.

Type `Region`

`selected`

Selects/deselects the chunk.

Type boolean

`sensors`

List of sensors in the chunk.

Type `Sensors`

`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 Added chunk.

Return type `Chunk`

`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.`**`CoordinateSystem`**

Provides access to geographic coordinate systems

```python
import PhotoScan

chunk = PhotoScan.Chunk()

crs = PhotoScan.CoordinateSystem()
crs.init("EPSG::32641")

gc = chunk.ground_control
gc.projection = crs
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.**DenseCloud**

Dense cloud instance

copy()

Returns a copy of the dense cloud.

Returns Copy of the dense cloud.

Return type `DenseCloud`

cropSelection()

Crops selected faces and free vertices from the mesh.

removeSelection()

Remove selected faces and free vertices from the mesh.

class PhotoScan.**DenseClouds**(*chunk*)

List of dense clouds in the chunk for each frame

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*], *compression = 6*, *absolute_paths = False*)

Saves document to the specified file.

Parameters

- **path** (*string*) – optional path to the file.
- **compression** (*int*) – project compression level.
- **absolute_paths** (*boolean*) – store absolute image paths.

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

meta

Document meta data.

Type MetaData

path

Path to the document file.

Type string

class PhotoScan.**Frame**

Photo instance

alpha()

Returns alpha channel data.

Returns Alpha channel data.

Return type Image

copy()

Returns a copy of the photo.

Returns Copy of the photo.

Return type Photo

image()

Returns image data.

Returns Image data.

Return type Image

mask()

Returns mask data.

Returns Mask data.

Return type Image

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

setMask(*mask*)

Initializes mask from image data.

Parameters **mask** (Image) – Mask image.

Returns Success of operation.

Return type boolean

thumbnail()

Returns thumbnail data.

Returns Thumbnail data.

Return type `Image`

`height`
Image height.

Type int

`layer`
Layer index in the image file.

Type int

`meta`
Frame meta data.

Type `MetaData`

`path`
Path to the image file.

Type string

`width`
Image width.

Type int

class `PhotoScan.`**`Frames`**(*camera*)
Collection of frames for the camera

`append`(*photo*)
Appends a frame to the camera.

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

`insert`(*index*, *photo*)
Insert a frame to the camera.

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

`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.`**`CoordinateSystem`**

Provides access to geographic coordinate systems

`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’, ‘bramor’]

Returns Success of operation.

Return type boolean

loadExif()

Imports camera locations from EXIF meta data.

Returns Success of operation.

Return type boolean

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

crs

Ground control coordinate system.

Type CoordinateSystem

locations

Ground control coordinates.

Type GroundControlLocations

projection

Ground control coordinate system.

Type CoordinateSystem

class PhotoScan.**GroundControlLocation**(*chunk*, *item*)

Provides access to the ground control coordinates for the given photo or marker

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 camera or marker.

Parameters **item** (Camera or Marker) – Camera or Marker instance.

items()
List of items.

keys()
List of item keys.

values()
List of item values.

class PhotoScan.**Image**(*width*, *height*, *cn*, *format='U8'*)
1 or 3-channel image

copy()
Makes a copy of the image.

Returns copy of the image

Return type Image

load(*path*, *layer=0*, *format='U8'*)
Loads image from the file.

Parameters

- **path** (*string*) – path to the image file
- **format** (*string*) – pixel data type in ['U8', ‘F32']

Returns success of operation

Return type boolean

resize(*width*, *height*)
Resizes image to specified dimensions.

Parameters

- **width** (*int*) – new image width
- **height** (*int*) – new image height

Returns resized image

Return type Image

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

format
Data type used to store pixel values.

Type string

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

meta
Marker meta data.

Type `MetaData`

position
Marker position in the current frame.

Type `Vector`

positions
List of marker positions in each frame.

Type `MarkerPositions`

projections
List of marker projections.

Type MarkerProjections

selected
Selects/deselects the marker.

Type boolean

class PhotoScan.**PyMarkerPositions**(*marker*)
List of marker positions for each frame

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.**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`

`col`(*index*)

Returns column of the matrix.

Returns matrix column.

Return type `Vector`

`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`()

Returns an inverted copy of the matrix.

Returns inverted matrix.

Return type `Matrix`

`row`(*index*)

Returns row of the matrix.

Returns matrix row.

Return type `Vector`

`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.**MetaData**(*object*)
Collection of object properties

items()
List of items.

keys()
List of item keys.

values()
List of item values.

class PhotoScan.**Model**
Triangular mesh model instance

area()
Returns area of the model surface.

Returns Model area.

Return type float

closeHoles(*level = 30*)
Fills holes in the model surface.

Parameters **level** (*int*) – Hole size threshold in percents.

Returns Success of operation.

Return type boolean

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.

fixTopology()
Removes polygons causing topological problems.

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.

renderDepth(*transform*, *calibration*)
Renders model depth image for specified viewpoint.

Parameters

- **transform** (Matrix) – Camera location.
- **calibration** (Calibration) – Camera calibration.

Returns Rendered image.

Return type Image

renderImage (*transform*, *calibration*)

Renders model image for specified viewpoint.

Parameters

- **transform** (Matrix) – Camera location.
- **calibration** (Calibration) – Camera calibration.

Returns Rendered image.

Return type Image

renderMask (*transform*, *calibration*)

Renders model mask image for specified viewpoint.

Parameters

- **transform** (Matrix) – Camera location.
- **calibration** (Calibration) – Camera calibration.

Returns Rendered image.

Return type Image

save (*path*, *binary=True*, *precision=6*, *texture_format='jpg'*, *texture=True*, *normals=True*, *colors=True*, *cameras=True*[, *comment*][, *format*][, *projection*][, *shift*])

Exports generated model for the chunk.

Parameters

- **path** (*string*) – Path to output model.
- **binary** (*boolean*) – Enables/disables binary encoding (if supported by format).
- **precision** (*int*) – Number of digits afer the decimal point (for text formats).
- **texture_format** (*string*) – Texture format in ['jpg', 'png', 'tif', 'exr', 'bmp'].
- **texture** (*boolean*) – Enables/disables texture export.
- **normals** (*boolean*) – Enables/disables export of vertex normals.
- **colors** (*boolean*) – Enables/disables export of vertex colors.
- **cameras** (*boolean*) – Enables/disables camera export.
- **comment** (*string*) – Optional comment (if supported by selected format).
- **format** (*string*) – Export format in ['3ds', 'obj', 'ply', 'vrml', 'collada', 'dxf', 'fbx', 'pdf', 'u3d', 'kmz'].
- **projection** (CoordinateSystem) – Output coordinate system.
- **shift** (*3-element vector*) – Optional shift to be applied to vertex coordinates.

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

setTexture(*image*, *page=0*)

Initializes texture from image data.

Parameters

- **image** (Image) – Texture image.
- **page** (*int*) – Texture index for multitextured models.

Returns Success of operation.

Return type boolean

texture(*page=0*)

Returns texture image.

Parameters **page** (*int*) – Texture index for multitextured models.

Returns Texture image.

Return type Image

volume()

Returns volume of the closed model surface.

Returns Model volume.

Return type float

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.**Models**(*chunk*)

List of models in the chunk for each frame

class PhotoScan.**Camera**

Camera instance

alpha()

Returns alpha channel data.

Returns Alpha channel data.

Return type Image

append(*path*[, *layer*])

Appends a new frame to the camera.

Parameters

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

Returns Success of operation.

Return type boolean

copy()

Returns a copy of the photo.

Returns Copy of the photo.

Return type `Photo`

image()

Returns image data.

Returns Image data.

Return type `Image`

insert(*index*, *path*[, *layer*])

Inserts a new frame to the camera.

Parameters

- **index** (*int*) – Position in the list of frames where the new frame should be inserted.
- **path** (*string*) – Path to the image file to be loaded.
- **layer** (*int*) – Optional layer index in case of multipage files.

Returns Success of operation.

Return type boolean

mask()

Returns mask data.

Returns Mask data.

Return type `Image`

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

Loads specified image file.

Parameters

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

setMask (*mask*)
Initializes mask from image data.

Parameters **mask** (`Image`) – Mask image.

Returns Success of operation.

Return type boolean

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

frames
Camera frames.

Type `Photos`

height
Image height.

Type int

label
Camera label.

Type string

layer
Layer index in the image file.

Type int

meta
Camera meta data.

Type `MetaData`

path
Path to the image file.

Type string

selected
Selects/deselects the photo.

Type boolean

sensor
Camera sensor.

Type `Sensor`

transform
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.`**`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 `CoordinateSystem`) – 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.`**`PointCloudCameras`**(*point_cloud*)

Collection of `PointCloudProjections` objects indexed by corresponding cameras

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

`frame`

Frame index.

Type int

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*, *camera*)
Collection of `PointCloudProjection` for the camera

class `PhotoScan.`**`Region`**
Region parameters

center
Region center coordinates.

Type `Vector`

rot
Region rotation matrix.

Type `Matrix`

size
Region size.

Type `Vector`

class `PhotoScan.`**`Scalebar`**
Scalebar instance

copy()
Returns a copy of the scalebar.

Returns Copy of the scalebar.

Return type `Scalebar`

label
Scalebar label.

Type string

meta
Scalebar meta data.

Type `MetaData`

selected
Selects/deselects the scalebar.

Type boolean

class PhotoScan.**Scalebars**(*chunk*)
Collection of scalebars in the chunk

add(*scalebar=None*)
Adds new marker to the chunk.

Parameters **scalebar** (Scalebar) – Optional argument specifying the scalebar to be added. An empty scalebar is added if unspecified.

Returns Index of the added scalebar.

Return type int

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

Parameters **scalebar** (Scalebar) – Scalebar to be looked for.

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

Return type int

remove(*scalebar*)
Removes specified scalebar from the chunk.

Parameters **scalebar** (Scalebar or int) – Scalebar object to be removed or index in the list of scalebars.

Returns Success of operation.

Return type boolean

class PhotoScan.**Sensor**
Sensor instance

copy()
Returns a copy of the photo.

Returns Copy of the photo.

Return type Photo

calibration
Refined calibration of the photo.

Type Calibration

fixed
Fix calibration flag.

Type boolean

focal_length
Focal length in mm.

Type float

height
Image height.

Type int

label
Camera label.

Type string

pixel_height
Pixel height in mm.

Type float

pixel_width
Pixel width in mm.

Type float

type
Sensor projection model in ['frame', 'spherical'].

Type string

user_calib
Custom calibration used as initial calibration during photo alignment.

Type Calibration

width
Image width.

Type int

class PhotoScan.**Sensors**(*chunk*)
Collection of sensors in the chunk

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

Parameters **sensor** (Sensor) – Sensor object.

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.**Utils**
Utility functions.

createDifferenceMask(*image*, *background*, *tolerance=10*, *fit_colors=True*)
Creates mask from a pair of images or an image and specified color.

Parameters

- **image** (Image) – Image to be masked.
- **background** (Image or color tuple) – Background image or color value.
- **tolerance** (*int*) – Tolerance value.
- **fit_colors** (*boolean*) – Enables white balance correction.

Returns Resulting mask.

Return type Image

estimateImageQuality(*image*)

Estimates image sharpness.

Parameters **image** (Image) – Image to be analyzed.

Returns Quality metric.

Return type float

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

norm()

Returns norm of the 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`

CHAPTER

THREE

PYTHON API CHANGE LOG

3.1 PhotoScan version 1.0.0 build 1795

- Added DenseCloud and DenseClouds classes
- Added Chunk.exportModel() and Chunk.importModel() methods
- Added Chunk.estimateImageQuality() method
- Added Photo.thumbnail() method
- Added Image.resize() method
- Added Camera.meta, Marker.meta, Scalebar.meta and Photo.meta attributes
- Added Chunk.dense_cloud and Chunk.dense_clouds attributes
- Added page parameter to Model.setTexture() and Model.texture() methods

3.2 PhotoScan version 1.0.0 build 1742

- Added Chunk.buildDenseCloud() and Chunk.smoothModel() methods
- Added Application.enumOpenCLDevices() method
- Added Utils.estimateImageQuality() method
- Removed Chunk.buildDepth() method
- Removed Camera.depth() and Camera.setDepth() methods
- Removed Frame.depth() and Frame.setDepth() methods
- Removed Frame.depth_calib attribute
- Changed parameters of Chunk.buildModel() and Chunk.buildTexture() methods
- Changed parameters of Chunk.exportPoints() method
- Changed parameters of Model.save() method
- Changed return value of Chunks.add() method
- Added shortcut parameter to Application.addMenuItem() method
- Added absolute_paths parameter to Document.save() method
- Added fit_f, fit_cxcy, fit_k1k2k3 and fit_k4 parameters to Chunk.optimizePhotos() method

3.3 PhotoScan version 0.9.1 build 1703

- Added Sensor class
- Added Scalebar class
- Added Camera.sensor attribute
- Added Chunk.sensors attribute
- Added Calibration.width and Calibration.height attributes
- Added Chunk.refineMatches() method
- Added Model.area() and Model.volume() methods
- Added Model.renderDepth(), Model.renderImage() and Model.renderMask() methods
- Added MetaData class
- Added Chunk.meta and Document.meta attributes
- Added Calibration.project() and Calibration.unproject() methods
- Added Calibration.k4 attribute
- Added Application.addMenuItem() method
- Added Model.closeHoles() and Model.fixTopology() methods

3.4 PhotoScan version 0.9.0 build 1586

- Added Camera class
- Added Frame class
- Added CoordinateSystem class
- Removed Photo class (deprecated)
- Removed GeoProjection class (deprecated)
- Added Chunk.exportReport() method
- Added Chunk.trackMarkers() and Chunk.detectMarkers() methods
- Added Chunk.extractFrames() and Chunk.removeFrames() methods
- Added Chunk.matchPhotos() method
- Added Chunk.buildDepth() method
- Added Chunk.resetDepth() method
- Revised Chunk.alignPhotos() method
- Revised Chunk.buildPoints() method
- Revised Chunk.buildModel() method
- Added Chunk.cameras property
- Removed Chunk.photos property (deprecated)
- Added Utils.createDifferenceMask() method

3.5 PhotoScan version 0.8.5 build 1423

- 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

3.6 PhotoScan version 0.8.4 build 1289

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

3.7 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

3.8 PhotoScan version 0.8.3 build 1154

Initial version of PhotoScan Python API

PYTHON MODULE INDEX