Metashape Python Reference

Release 2.1.0

Agisoft LLC

Dec 30, 2023

CONTENTS

CHAPTER

ONE

OVERVIEW

1.1 Introduction to Python scripting in Metashape Professional

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

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

Metashape Professional uses Python 3.8 as a scripting engine.

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

- From Metashape "Console" pane using it as standard Python console.
- From the "Tools" menu using "Run script…" command.
- From command line using "-r" argument and passing the path to the script as an argument.

The following Metashape funtionality can be accessed from Python scripts:

- Open/save/create Metashape 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.
- Start and control network processing tasks.

CHAPTER

TWO

APPLICATION MODULES

Metashape module provides access to the core processing functionality, including support for inspection and manipulation with project data.

The main component of the module is a Document class, which represents a Metashape project. Multiple Document instances can be created simultaneously if needed. Besides that a currently opened project in the application can be accessed using Metashape.app.document property.

The following example performs main processing steps on existing project and saves back the results:

```
>>> import Metashape
>>> doc = Metashape.app.document
>>> doc.open("project.psz")
>>> chunk = doc.chunk
>>> chunk.matchPhotos(downscale=1, generic_preselection=True, reference_
↪preselection=False)
>>> chunk.alignCameras()
>>> chunk.buildDepthMaps(downscale=4, filter_mode=Metashape.AggressiveFiltering)
>>> chunk.buildModel(source_data=Metashape.DepthMapsData, surface_type=Metashape.
↪Arbitrary, interpolation=Metashape.EnabledInterpolation)
>>> chunk.buildUV(mapping_mode=Metashape.GenericMapping)
>>> chunk.buildTexture(blending_mode=Metashape.MosaicBlending, texture_size=4096)
>>> doc.save()
```

class Metashape.**Antenna**

GPS antenna position relative to camera.

copy()

Return a copy of the object.

Returns

A copy of the object.

Return type

Metashape.Antenna

fixed

Fix antenna flag.

Type

bool

location

Antenna coordinates.

Type
Metashape.Vector

`location_acc`

Antenna location accuracy.

Type
Metashape.Vector

`location_covariance`

Antenna location covariance.

Type
Metashape.Matrix

`location_ref`

Antenna location reference.

Type
Metashape.Vector

`rotation`

Antenna rotation angles.

Type
Metashape.Vector

`rotation_acc`

Antenna rotation accuracy.

Type
Metashape.Vector

`rotation_covariance`

Antenna rotation covariance.

Type
Metashape.Matrix

`rotation_ref`

Antenna rotation reference.

Type
Metashape.Vector

class Metashape.Application

Application class provides access to several global application attributes, such as document currently loaded in the user interface, software version and GPU device configuration. It also contains helper routines to prompt the user to input various types of parameters, like displaying a file selection dialog or coordinate system selection dialog among others.

An instance of Application object can be accessed using Metashape.app attribute, so there is usually no need to create additional instances in the user code.

The following example prompts the user to select a new coordinate system, applies it to the ative chunk and saves the project under the user selected file name:

```
>>> import Metashape
>>> doc = Metashape.app.document
>>> crs = Metashape.app.getCoordinateSystem("Select Coordinate System", doc.chunk.
```

(continues on next page)

(continued from previous page)

```
↪crs)
>>> doc.chunk.crs = crs
>>> path = Metashape.app.getSaveFileName("Save Project As")
>>> try:
...     doc.save(path)
... except RuntimeError:
...     Metashape.app.messageBox("Can't save project")
```

class ConsolePane

ConsolePane class provides access to the console pane

clear()

Clear console pane.

contents

Console pane contents.

Type
str

class ModelView

ModelView class provides access to the model view

class ModelViewMode

Model view mode in [ModelViewTextured, ModelViewShaded, ModelViewSolid, ModelViewWireframe, ModelViewElevation, ModelViewConfidence]

class PointCloudViewMode

Point cloud view mode in [PointCloudViewSolid, PointCloudViewColor, PointCloudViewClassification, PointCloudViewIntensity, PointCloudViewElevation, PointCloudViewConfidence, PointCloudViewReturnNumber, PointCloudViewScanAngle, PointCloudViewSourceId]

class TiePointsViewMode

Tie points view mode in [TiePointsViewColor, TiePointsViewElevation, TiePointsViewVariance]

class TiledModelViewMode

Tiled model view mode in [TiledModelViewTextured, TiledModelViewSolid, TiledModelViewWireframe, TiledModelViewElevation]

captureVideo(*path*, *width*, *height*[, *frame_rate*][, *transparent*][, *compressed*][, *hide_items*])

Capture video using camera track. Transparent capture can't be compressed.Method requires gui and inaccessible from python module. If script is passed as a program argument, –gui flag should be specified.

Parameters
- **path** (*str:arg width: Video width.*) – Output path.
- **height** (*int*) – Video height.
- **frame_rate** (*int*) – Video frame rate.
- **transparent** (*bool*) – Sets transparent background.
- **compressed** (*bool*) – Enables video compression.
- **hide_items** (*bool*) – Hides all items.

captureView([*width*][, *height*][, *transparent*][, *hide_items*])

Capture image from model view.

Parameters
- **width** (*int*) – Image width.
- **height** (*int*) – Image height.

- **transparent** (*bool*) – Sets transparent background.
- **hide_items** (*bool*) – Hides all items.

Returns
Captured image.

Return type
Metashape.Image

model_view_mode

Model view mode.

Type
Metashape.Application.ModelView.ModelViewMode

point_cloud_view_mode

Point cloud view mode.

Type
Metashape.Application.ModelView.PointCloudViewMode

texture_view_mode

Texture view mode.

Type
Metashape.Model.TextureType

tie_points_view_mode

Tie points view mode.

Type
Metashape.Application.ModelView.TiePointsViewMode

tiled_model_view_mode

Tiled model view mode.

Type
Metashape.Application.ModelView.TiledModelViewMode

view_mode

View mode.

Type
Metashape.DataSource

viewpoint

Viewpoint in the model view.

Type
Metashape.Viewpoint

class OrthoView

OrthoView class provides access to the ortho view

captureView([*width*][, *height*][, *transparent*][, *hide_items*])

Capture image from ortho view.

Parameters

- **width** (*int*) – Image width.
- **height** (*int*) – Image height.
- **transparent** (*bool*) – Sets transparent background.
- **hide_items** (*bool*) – Hides all items.

Returns
Captured image.

Return type
Metashape.Image

view_mode

View mode.

Type

Metashape.DataSource

class PhotosPane

PhotosPane class provides access to the photos pane

resetFilter()

Reset photos pane filter.

setFilter(*items*)

Set photos pane filter.

Parameters

items (*list[Metashape.Camera | Metashape.Marker]*) – filter to apply.

class Settings

PySettings()

Application settings

language

User interface language.

Type

str

load()

Load settings from disk.

log_enable

Enable writing log to file.

Type

bool

log_path

Log file path.

Type

str

network_enable

Network processing enabled flag.

Type

bool

network_host

Network server host name.

Type

str

network_path

Network data root path.

Type

str

network_port

Network server control port.

Type

int

project_absolute_paths

Store absolute image paths in project files.

Type
bool

project_compression

Project compression level.

Type
int

save()

Save settings on disk.

setValue(*key*, *value*)

Set settings value.

Parameters
- **key** (*str*) – Key.
- **value** (*object*) – Value.

value(*key*)

Return settings value.

Parameters
key (*str*) – Key.

Returns
Settings value.

Return type
object

activated

Metashape activation status.

Type
bool

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

Create a new menu entry.

Parameters
- **label** (*str*) – Menu item label.
- **func** (*function*) – Function to be called.
- **shortcut** (*str*) – Keyboard shortcut.
- **icon** (*str*) – Icon.

addMenuSeparator(*label*)

Add menu separator.

Parameters
label (*str*) – Menu label.

console_pane

Console pane.

Type
Metashape.Application.ConsolePane

cpu_enable

Use CPU when GPU is active.

Type
bool

document

Main application document object.

Type
Metashape.Document

enumGPUDevices()

Enumerate installed GPU devices.

Returns
A list of devices.

Return type
list

getBool(*label=''*)

Prompt user for the boolean value.

Parameters
label (*str*) – Optional text label for the dialog.

Returns
Boolean value selected by the user.

Return type
bool

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

Prompt user for coordinate system.

Parameters

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

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

Return type
Metashape.CoordinateSystem

getExistingDirectory([*hint*][, *dir*])

Prompt user for the existing folder.

Parameters

- **hint** (*str*) – Optional text label for the dialog.
- **dir** (*str*) – Optional default folder.

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

Return type
str

getFloat(*label=''*, *value=0*)

Prompt user for the floating point value.

Parameters

- **label** (*str*) – 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*)

Prompt user for the integer value.

Parameters

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

Returns

Integer value entered by the user.

Return type

int

getOpenFileName([*hint*][, *dir*][, *filter*])

Prompt user for the existing file.

Parameters

- **hint** (*str*) – Optional text label for the dialog.
- **dir** (*str*) – Optional default folder.
- **filter** (*str*) – Optional file filter, e.g. "Text file (*.txt)" or "*.txt". Multiple filters are separated with ";;".

Returns

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

Return type

str

getOpenFileNames([*hint*][, *dir*][, *filter*])

Prompt user for one or more existing files.

Parameters

- **hint** (*str*) – Optional text label for the dialog.
- **dir** (*str*) – Optional default folder.
- **filter** (*str*) – Optional file filter, e.g. "Text file (*.txt)" or "*.txt". Multiple filters are separated with ";;".

Returns

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

Return type

list

getSaveFileName([*hint*][, *dir*][, *filter*])

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

Parameters

- **hint** (*str*) – Optional text label for the dialog.
- **dir** (*str*) – Optional default folder.
- **filter** (*str*) – Optional file filter, e.g. "Text file (*.txt)" or "*.txt". Multiple filters are separated with ";;".

Returns

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

Return type

str

getString(*label=''*, *value=''*)

Prompt user for the string value.

Parameters

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

Returns

String entered by the user.

Return type

str

gpu_mask

GPU device bit mask: 1 - use device, 0 - do not use (i.e. value 5 enables device number 0 and 2).

Type

int

messageBox(*message*)

Display message box to the user.

Parameters

message (*str*) – Text message to be displayed.

model_view

Model view.

Type

Metashape.Application.ModelView

ortho_view

Ortho view.

Type

Metashape.Application.OrthoView

photos_pane

Photos pane.

Type

Metashape.Application.PhotosPane

quit()

Exit application.

releaseFreeMemory()

Call malloc_trim on Linux (does nothing on other OS).

removeMenuItem(*label*)

Remove menu entry with given label (if exists). If there are multiple entries with given label - all of them will be removed.

Parameters
`label` (*str*) – Menu item label.

settings

Application settings.

Type
Metashape.Application.Settings

title

Application name.

Type
str

update()

Update user interface during long operations.

version

Metashape version.

Type
str

class Metashape.AttachedGeometry

Attached geometry data.

GeometryCollection(*geometries*)

Create a GeometryCollection geometry.

Parameters
`geometries` (*list[Metashape.AttachedGeometry]*) – Child geometries.

Returns
A GeometryCollection geometry.

Return type
Metashape.AttachedGeometry

LineString(*coordinates*)

Create a LineString geometry.

Parameters
`coordinates` (*list[int]*) – List of vertex coordinates.

Returns
A LineString geometry.

Return type
Metashape.AttachedGeometry

MultiLineString(*geometries*)

Create a MultiLineString geometry.

Parameters

geometries (*list[Metashape.AttachedGeometry]*) – Child line strings.

Returns

A point geometry.

Return type

Metashape.AttachedGeometry

MultiPoint(*geometries*)

Create a MultiPoint geometry.

Parameters

geometries (*list[Metashape.AttachedGeometry]*) – Child points.

Returns

A point geometry.

Return type

Metashape.AttachedGeometry

MultiPolygon(*geometries*)

Create a MultiPolygon geometry.

Parameters

geometries (*list[Metashape.AttachedGeometry]*) – Child polygons.

Returns

A point geometry.

Return type

Metashape.AttachedGeometry

Point(*key*)

Create a Point geometry.

Parameters

key (*int*) – Point marker key.

Returns

A point geometry.

Return type

Metashape.AttachedGeometry

Polygon(*exterior_ring*[, *interior_rings*])

Create a Polygon geometry.

Parameters

- **exterior_ring** (*list[int]*) – Point coordinates.
- **interior_rings** (*list[int]*) – Point coordinates.

Returns

A Polygon geometry.

Return type

Metashape.AttachedGeometry

coordinates

List of vertex keys.

Type
list[int]

geometries

List of child geometries.

Type
list[*Metashape.AttachedGeometry*]

type

Geometry type.

Type
Metashape.Geometry.Type

class Metashape.BBox

Axis aligned bounding box

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.BBox

max

Maximum bounding box extent.

Type
Metashape.Vector

min

Minimum bounding box extent.

Type
Metashape.Vector

size

Bounding box dimension.

Type
int

class Metashape.BlendingMode

Blending mode in [AverageBlending, MosaicBlending, MinBlending, MaxBlending, DisabledBlending]

class Metashape.Calibration

Calibration object contains camera calibration information including image size, focal length, principal point coordinates and distortion coefficients.

b1

Affinity.

Type
float

b2

Non-orthogonality.

Type
float

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.Calibration

covariance_matrix

Covariance matrix.

Type
Metashape.Matrix

covariance_params

Covariance matrix parameters.

Type
list[str]

cx

Principal point X coordinate.

Type
float

cy

Principal point Y coordinate.

Type
float

error(*point*, *proj*)

Return projection error.

Parameters

- **point** (`Metashape.Vector`) – Coordinates of the point to be projected.
- **proj** (`Metashape.Vector`) – Pixel coordinates of the point.

Returns
2D projection error.

Return type
Metashape.Vector

f

Focal length.

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

load(*path*, *format=CalibrationFormatXML*)

Loads calibration from file.

Parameters

- **path** (*str*) – path to calibration file
- **format** (`Metashape.CalibrationFormat`) – Calibration format.

p1

Decentering distortion coefficient P1.

Type
float

p2

Decentering distortion coefficiant P2.

Type
float

p3

Decentering distortion coefficient P3.

Type
float

p4

Decentering distortion coefficiant P4.

Type
float

project(*point*)

Return projected pixel coordinates of the point.

Parameters

point (Metashape.Vector) – Coordinates of the point to be projected.

Returns

2D projected point coordinates.

Return type

Metashape.Vector

rpc

RPC model.

Type

Metashape.RPCModel

save(*path*, *format=CalibrationFormatXML*[, *label*][, *pixel_size*][, *focal_length*], *cx = 0*, *cy = 0*)

Saves calibration to file.

Parameters

- **path** (*str*) – path to calibration file
- **format** (Metashape.CalibrationFormat) – Calibration format.
- **label** (*str*) – Calibration label used in Australis, CalibCam and CalCam formats.
- **pixel_size** (Metashape.Vector) – Pixel size in mm used to convert normalized calibration coefficients to Australis and CalibCam coefficients.
- **focal_length** (*float*) – Focal length (Grid calibration format only).
- **cx** (*float*) – X principal point coordinate (Grid calibration format only).
- **cy** (*float*) – Y principal point coordinate (Grid calibration format only).

type

Camera model.

Type

Metashape.Sensor.Type

unproject(*point*)

Return direction corresponding to the image point.

Parameters

point (Metashape.Vector) – Pixel coordinates of the point.

Returns

3D vector in the camera coordinate system.

Return type

Metashape.Vector

width

Image width.

Type

int

class Metashape.CalibrationFormat

Calibration format in [CalibrationFormatXML, CalibrationFormatAustralis, CalibrationFormatAustralisV7, CalibrationFormatPhotoModeler, CalibrationFormatCalibCam, CalibrationFormatCalCam, CalibrationFormatInpho, CalibrationFormatUSGS, CalibrationFormatPix4D, CalibrationFormatOpenCV, CalibrationFormatPhotomod, CalibrationFormatGrid, CalibrationFormatSTMap]

class Metashape.Camera

Camera instance

```
>>> import Metashape
>>> chunk = Metashape.app.document.addChunk()
>>> chunk.addPhotos(["IMG_0001.jpg", "IMG_0002.jpg"])
>>> camera = chunk.cameras[0]
>>> camera.photo.meta["Exif/FocalLength"]
'18'
```

The following example describes how to create multispectal camera layout:

```
>>> import Metashape
>>> doc = Metashape.app.document
>>> chunk = doc.chunk
>>> rgb = ["RGB_0001.JPG", "RGB_0002.JPG", "RGB_0003.JPG"]
>>> nir = ["NIR_0001.JPG", "NIR_0002.JPG", "NIR_0003.JPG"]
>>> images = [rgb[0], nir[0], rgb[1], nir[1], rgb[2], nir[2]]
>>> groups = [2, 2, 2]
>>> chunk.addPhotos(filenames=images, filegroups=groups, layout=Metashape.
↪MultiplaneLayout)
```

class Reference

Camera reference data.

accuracy

Camera location accuracy.

Type
Metashape.Vector

enabled

Location enabled flag.

Type
bool

location

Camera coordinates.

Type
Metashape.Vector

location_accuracy

Camera location accuracy.

Type
Metashape.Vector

location_enabled

Location enabled flag.

Type
bool

rotation

Camera rotation angles.

Type

Metashape.Vector

rotation_accuracy

Camera rotation accuracy.

Type

Metashape.Vector

rotation_enabled

Rotation enabled flag.

Type

bool

class Type

Camera type in [Regular, Keyframe]

calibration

Adjusted camera calibration including photo-invariant parameters.

Type

Metashape.Calibration

center

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

Type

Metashape.Vector

chunk

Chunk the camera belongs to.

Type

Metashape.Chunk

component

Camera component.

Type

Metashape.Component

enabled

Enables/disables the photo.

Type

bool

error(*point*, *proj*)

Returns projection error.

Parameters

- **point** (`Metashape.Vector`) – Coordinates of the point to be projected.
- **proj** (`Metashape.Vector`) – Pixel coordinates of the point.

Returns

2D projection error.

Return type
Metashape.Vector

`frames`
Camera frames.

Type
list[*Metashape.Camera*]

`group`
Camera group.

Type
Metashape.CameraGroup

`image()`
Returns image data.

Returns
Image data.

Return type
Metashape.Image

`key`
Camera identifier.

Type
int

`label`
Camera label.

Type
str

`layer_index`
Camera layer index.

Type
int

`location_covariance`
Camera location covariance.

Type
Metashape.Matrix

`mask`
Camera mask.

Type
Metashape.Mask

`master`
Master camera.

Type
Metashape.Camera

meta

Camera meta data.

Type

Metashape.MetaData

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

Loads specified image file.

Parameters

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

orientation

Image orientation (1 - normal, 6 - 90 degree, 3 - 180 degree, 8 - 270 degree).

Type

int

photo

Camera photo.

Type

Metashape.Photo

planes

Camera planes.

Type

list[*Metashape.Camera*]

project(*point*)

Returns coordinates of the point projection on the photo.

Parameters

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

Returns

2D point coordinates.

Return type

Metashape.Vector

reference

Camera reference data.

Type

Metashape.Camera.Reference

rotation_covariance

Camera rotation covariance.

Type

Metashape.Matrix

selected

Selects/deselects the photo.

Type

bool

sensor

Camera sensor.

Type
Metashape.Sensor

shutter

Camera shutter.

Type
Metashape.Shutter

thumbnail

Camera thumbnail.

Type
Metashape.Thumbnail

transform

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

Type
Metashape.Matrix

type

Camera type.

Type
Metashape.Camera.Type

unproject(*point*)

Returns coordinates of the point which will have specified projected coordinates.

Parameters
point (`Metashape.Vector`) – Projection coordinates.

Returns
3D point coordinates.

Return type
Metashape.Vector

vignetting

Vignetting for each band.

Type
list[*Metashape.Vignetting*]

class Metashape.CameraGroup

CameraGroup objects define groups of multiple cameras. The grouping is established by assignment of a CameraGroup instance to the Camera.group attribute of participating cameras.

The type attribute of CameraGroup instances defines the effect of such grouping on processing results and can be set to Folder (no effect) or Station (coincident projection centers).

class Type

Camera group type in [Folder, Station]

label

Camera group label.

Type
str

selected
Current selection state.

Type
bool

type
Camera group type.

Type
Metashape.CameraGroup.Type

class Metashape.CameraTrack
Camera track.

chunk
Chunk the camera track belongs to.

Type
Metashape.Chunk

duration
Animation duration.

Type
float

field_of_view
Vertical field of view in degrees.

Type
float

interpolate(*time*)
Get animation camera transform matrix.

Parameters
time (*float*) – Animation time point.

Returns
Interpolated camera transformation matrix in chunk coordinate system.

Return type
Metashape.Matrix

keyframes
Camera track keyframes.

Type
list[*Metashape.Camera*]

label
Animation label.

Type
str

load(*path*[, *projection*])

Load camera track from file.

Parameters

- **path** (*str*) – Path to camera track file
- **projection** (Metashape.CoordinateSystem) – Camera track coordinate system.

loop

Loop track.

Type

bool

meta

Camera track meta data.

Type

Metashape.MetaData

save(*path*[, *file_format*][, *drone_name*][, *payload_name*][, *payload_position*][, *max_waypoints*][, *projection*])

Save camera track to file.

Parameters

- **path** (*str*) – Path to camera track file
- **file_format** (*str*) – File format. "deduce": - Deduce from extension, "path": Path, "earth": Google Earth KML, "pilot": DJI Pilot KML, "wpml": DJI WPML KMZ, "trinity": Asctec Trinity CSV, "autopilot": Asctec Autopilot CSV, "litchi": Litchi CSV
- **drone_name** (*str*) – Drone model. "M300 RTK": - DJI Matrice 300 RTK, "M30": - DJI Matrice 30, "M30T": - DJI Matrice 30T, "M3E": - DJI Mavic 3E, "M3T": - DJI Mavic 3T
- **payload_name** (*str*) – Payload model. "P1 24mm": - DJI Zenmuse P1 (24 mm lens), "P1 35mm": - DJI Zenmuse P1 (35 mm lens), "P1 50mm": - DJI Zenmuse P1 (50 mm lens), "H20": - DJI Zenmuse H20, "H20T": - DJI Zenmuse H20T, "H20N": - DJI Zenmuse H20N, "L1": - DJI Zenmuse L1, "M30": - DJI M30, "M30T": - DJI M30T, "M3E": - DJI Mavic 3E Camera, "M3T": - DJI Mavic 3T Camera
- **payload_position** (*str*) – Payload position. For M300 RTK drone: "Front left", "Front right", "Top". For other drones: "Main gimbal"
- **max_waypoints** (*int*) – Max waypoints per flight
- **projection** (Metashape.CoordinateSystem) – Camera track coordinate system.

smooth

Smooth path.

Type

bool

class Metashape.CamerasFormat

Camera orientation format in [CamerasFormatXML, CamerasFormatCHAN, CamerasFormatBoujou, CamerasFormatBundler, CamerasFormatOPK, CamerasFormatPATB, CamerasFormatBINGO, CamerasFormatORIMA, CamerasFormatAeroSys, CamerasFormatInpho, CamerasFormatSummit, CamerasFormatBlocksExchange, CamerasFormatRZML, CamerasFormatVisionMap, CamerasFormatABC, CamerasFormatFBX, CamerasFormatNVM, CamerasFormatMA]

class Metashape.**Chunk**

A Chunk object:

- provides access to all chunk components (sensors, cameras, camera groups, markers, scale bars)
- contains data inherent to individual frames (tie points, model, etc)
- implements processing methods (matchPhotos, alignCameras, buildPointCloud, buildModel, etc)
- provides access to other chunk attributes (transformation matrix, coordinate system, meta-data, etc..)

New components can be created using corresponding addXXX methods (addSensor, addCamera, addCameraGroup, addMarker, addScalebar, addFrame). Removal of components is supported by a single remove method, which can accept lists of various component types.

In case of multi-frame chunks the Chunk object contains an additional reference to the particular chunk frame, initialized to the current frame by default. Various methods that work on a per frame basis (matchPhotos, buildModel, etc) are applied to this particular frame. A frames attribute can be used to obtain a list of Chunk objects that reference all available frames.

The following example performs image matching and alignment for the active chunk:

```
>>> import Metashape
>>> chunk = Metashape.app.document.chunk
>>> for frame in chunk.frames:
...     frame.matchPhotos(downscale=1)
>>> chunk.alignCameras()
```

addCamera([*sensor*])

Add new camera to the chunk.

Parameters
sensor (Metashape.Sensor) – Sensor to be assigned to this camera.

Returns
Created camera.

Return type
Metashape.Camera

addCameraGroup()

Add new camera group to the chunk.

Returns
Created camera group.

Return type
Metashape.CameraGroup

addCameraTrack()

Add new camera track to the chunk.

Returns
Created camera track.

Return type
Metashape.CameraTrack

addDepthMaps()

Add new depth maps set to the chunk.

Returns
Created depth maps set.

Return type
Metashape.DepthMaps

addElevation()

Add new elevation model to the chunk.

Returns
Created elevation model.

Return type
Metashape.Elevation

addFrame()

Add new frame to the chunk.

Returns
Created frame.

Return type
Metashape.Chunk

addFrames([*chunk*][, *frames*], *copy_depth_maps=True*, *copy_point_cloud=True*, *copy_model=True*, *copy_tiled_model=True*, *copy_elevation=True*, *copy_orthomosaic=True*[, *progress*])

Add frames from specified chunk.

Parameters

- **chunk** (*int*) – Chunk to copy frames from.
- **frames** (*list[int]*) – List of frame keys to copy.
- **copy_depth_maps** (*bool*) – Copy depth maps.
- **copy_point_cloud** (*bool*) – Copy point cloud.
- **copy_model** (*bool*) – Copy model.
- **copy_tiled_model** (*bool*) – Copy tiled model.
- **copy_elevation** (*bool*) – Copy DEM.
- **copy_orthomosaic** (*bool*) – Copy orthomosaic.
- **progress** (*Callable[[float], None]*) – Progress callback.

addMarker([*point*], *visibility=False*)

Add new marker to the chunk.

Parameters

- **point** (`Metashape.Vector`) – Point to initialize marker projections.
- **visibility** (*bool*) – Enables visibility check during projection assignment.

Returns
Created marker.

Return type
Metashape.Marker

addMarkerGroup()

Add new marker group to the chunk.

Returns
Created marker group.

Return type
Metashape.MarkerGroup

addModel()

Add new model to the chunk.

Returns
Created model.

Return type
Metashape.Model

addOrthomosaic()

Add new orthomosaic to the chunk.

Returns
Created orthomosaic.

Return type
Metashape.Orthomosaic

addPhotos([*filenames*][, *filegroups*], *layout=UndefinedLayout*[, *group*], *strip_extensions=True*, *load_reference=True*, *load_xmp_calibration=True*, *load_xmp_orientation=True*, *load_xmp_accuracy=False*, *load_xmp_antenna=True*, *load_rpc_txt=False*[, *progress*])

Add a list of photos to the chunk.

Parameters

- **filenames** (*list[str]*) – List of files to add.
- **filegroups** (*list[int]*) – List of file groups.
- **layout** (`Metashape.ImageLayout`) – Image layout.
- **group** (*int*) – Camera group key.
- **strip_extensions** (*bool*) – Strip file extensions from camera labels.
- **load_reference** (*bool*) – Load reference coordinates.
- **load_xmp_calibration** (*bool*) – Load calibration from XMP meta data.
- **load_xmp_orientation** (*bool*) – Load orientation from XMP meta data.
- **load_xmp_accuracy** (*bool*) – Load accuracy from XMP meta data.
- **load_xmp_antenna** (*bool*) – Load GPS/INS offset from XMP meta data.
- **load_rpc_txt** (*bool*) – Load satellite RPC data from auxiliary TXT files.
- **progress** (*Callable[[float], None]*) – Progress callback.

addPointCloud()

Add new point cloud to the chunk.

Returns
Created point cloud.

Return type
Metashape.PointCloud

addPointCloudGroup()

Add new point cloud group to the chunk.

Returns
Created point cloud group.

Return type
Metashape.PointCloudGroup

addScalebar(*point1*, *point2*)

Add new scale bar to the chunk.

Parameters

- **point1** (`Metashape.Marker` | `Metashape.Camera`) – First endpoint.
- **point2** (`Metashape.Marker` | `Metashape.Camera`) – Second endpoint.

Returns
Created scale bar.

Return type
Metashape.Scalebar

addScalebarGroup()

Add new scale bar group to the chunk.

Returns
Created scale bar group.

Return type
Metashape.ScalebarGroup

addSensor([*source*])

Add new sensor to the chunk.

Parameters
source (`Metashape.Sensor`) – Sensor to copy parameters from.

Returns
Created sensor.

Return type
Metashape.Sensor

addTiledModel()

Add new tiled model to the chunk.

Returns
Created tiled model.

Return type
Metashape.TiledModel

alignCameras([*cameras*][, *point_clouds*], *min_image=2*, *adaptive_fitting=False*, *reset_alignment=False*, *subdivide_task=True*[, *progress*])

Perform photo alignment for the chunk.

Parameters

- **cameras** (*list[int]*) – List of cameras to align.
- **point_clouds** (*list[int]*) – List of point clouds to align.
- **min_image** (*int*) – Minimum number of point projections.
- **adaptive_fitting** (*bool*) – Enable adaptive fitting of distortion coefficients.
- **reset_alignment** (*bool*) – Reset current alignment.
- **subdivide_task** (*bool*) – Enable fine-level task subdivision.
- **progress** (*Callable[[float], None]*) – Progress callback.

analyzeImages([*cameras*], *filter_mask=False*[, *progress*])

Estimate image quality. Estimated value is stored in camera metadata with Image/Quality key. Cameras with quality less than 0.5 are considered blurred and we recommend to disable them.

Parameters

- **cameras** (*list[int]*) – List of cameras to be analyzed.
- **filter_mask** (*bool*) – Constrain analyzed image region by mask.
- **progress** (*Callable[[float], None]*) – Progress callback.

buildContours(*source_data=ElevationData*, *interval=1*, *min_value=-1e+10*, *max_value=1e+10*, *prevent_intersections=True*[, *progress*])

Build contours for the chunk.

Parameters

- **source_data** (Metashape.DataSource) – Source data for contour generation.
- **interval** (*float*) – Contour interval.
- **min_value** (*float*) – Minimum value of contour range.
- **max_value** (*float*) – Maximum value of contour range.
- **prevent_intersections** (*bool*) – Prevent contour intersections.
- **progress** (*Callable[[float], None]*) – Progress callback.

buildDem(*source_data=PointCloudData*, *interpolation=EnabledInterpolation*[, *projection*][, *region*][, *classes*], *flip_x=False*, *flip_y=False*, *flip_z=False*, *resolution=0*, *subdivide_task=True*, *workitem_size_tiles=10*, *max_workgroup_size=100*, *replace_asset=False*[, *frames*][, *progress*])

Build elevation model for the chunk.

Parameters

- **source_data** (Metashape.DataSource) – Selects between point cloud and tie points.
- **interpolation** (Metashape.Interpolation) – Interpolation mode.
- **projection** (Metashape.OrthoProjection) – Output projection.
- **region** (Metashape.BBox) – Region to be processed.
- **classes** (*list[int]*) – List of point classes to be used for surface extraction.
- **flip_x** (*bool*) – Flip X axis direction.
- **flip_y** (*bool*) – Flip Y axis direction.
- **flip_z** (*bool*) – Flip Z axis direction.
- **resolution** (*float*) – Output resolution in meters.

- **subdivide_task** (*bool*) – Enable fine-level task subdivision.
- **workitem_size_tiles** (*int*) – Number of tiles in a workitem.
- **max_workgroup_size** (*int*) – Maximum workgroup size.
- **replace_asset** (*bool*) – Replace default asset with generated DEM.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

buildDepthMaps(*downscale=4, filter_mode=MildFiltering*[, *cameras*], *reuse_depth=False, max_neighbors=16, subdivide_task=True, workitem_size_cameras=20, max_workgroup_size=100*[, *progress*])

Generate depth maps for the chunk.

Parameters

- **downscale** (*int*) – Depth map quality (1 - Ultra high, 2 - High, 4 - Medium, 8 - Low, 16 - Lowest).
- **filter_mode** (Metashape.FilterMode) – Depth map filtering mode.
- **cameras** (*list[int]*) – List of cameras to process.
- **reuse_depth** (*bool*) – Enable reuse depth maps option.
- **max_neighbors** (*int*) – Maximum number of neighbor images to use for depth map generation.
- **subdivide_task** (*bool*) – Enable fine-level task subdivision.
- **workitem_size_cameras** (*int*) – Number of cameras in a workitem.
- **max_workgroup_size** (*int*) – Maximum workgroup size.
- **progress** (*Callable[[float], None]*) – Progress callback.

buildModel(*surface_type=Arbitrary, interpolation=EnabledInterpolation, face_count=HighFaceCount, face_count_custom=200000, source_data=DepthMapsData*[, *classes*], *vertex_colors=True, vertex_confidence=True, volumetric_masks=False, keep_depth=True, replace_asset=False, split_in_blocks=False*[, *blocks_crs*], *blocks_size=250*[, *blocks_origin*], *clip_to_boundary=False, export_blocks=False, build_texture=True, output_folder='', trimming_radius=10*[, *cameras*][, *frames*], *subdivide_task=True, workitem_size_cameras=20, max_workgroup_size=100*[, *progress*])

Generate model for the chunk frame.

Parameters

- **surface_type** (Metashape.SurfaceType) – Type of object to be reconstructed.
- **interpolation** (Metashape.Interpolation) – Interpolation mode.
- **face_count** (Metashape.FaceCount) – Target face count.
- **face_count_custom** (*int*) – Custom face count.
- **source_data** (Metashape.DataSource) – Selects between point cloud, tie points, depth maps and laser scans.
- **classes** (*list[int]*) – List of point classes to be used for surface extraction.
- **vertex_colors** (*bool*) – Enable vertex colors calculation.
- **vertex_confidence** (*bool*) – Enable vertex confidence calculation.

- **volumetric_masks** (*bool*) – Enable strict volumetric masking.
- **keep_depth** (*bool*) – Enable store depth maps option.
- **replace_asset** (*bool*) – Replace default asset with generated model.
- **split_in_blocks** (*bool*) – Split model in blocks.
- **blocks_crs** (Metashape.CoordinateSystem) – Blocks grid coordinate system.
- **blocks_size** (*float*) – Blocks size in coordinate system units.
- **blocks_origin** (Metashape.Vector) – Blocks grid origin.
- **clip_to_boundary** (*bool*) – Clip to boundary shapes.
- **export_blocks** (*bool*) – Export completed blocks.
- **build_texture** (*bool*) – Generate preview textures.
- **output_folder** (*str*) – Path to output folder.
- **trimming_radius** (*int*) – Trimming radius (no trimming if zero).
- **cameras** (*list[int]*) – List of cameras to process.
- **frames** (*list[int]*) – List of frames to process.
- **subdivide_task** (*bool*) – Enable fine-level task subdivision.
- **workitem_size_cameras** (*int*) – Number of cameras in a workitem.
- **max_workgroup_size** (*int*) – Maximum workgroup size.
- **progress** (*Callable[[float], None]*) – Progress callback.

buildOrthomosaic(*surface_data=ModelData*, *blending_mode=MosaicBlending*, *fill_holes=True*, *ghosting_filter=False*, *cull_faces=False*, *refine_seamlines=False*[, *projection*][, *region*], *resolution=0*, *resolution_x=0*, *resolution_y=0*, *flip_x=False*, *flip_y=False*, *flip_z=False*, *subdivide_task=True*, *workitem_size_cameras=20*, *workitem_size_tiles=10*, *max_workgroup_size=100*, *replace_asset=False*[, *frames*][, *progress*])

Build orthomosaic for the chunk.

Parameters

- **surface_data** (Metashape.DataSource) – Orthorectification surface.
- **blending_mode** (Metashape.BlendingMode) – Orthophoto blending mode.
- **fill_holes** (*bool*) – Enable hole filling.
- **ghosting_filter** (*bool*) – Enable ghosting filter.
- **cull_faces** (*bool*) – Enable back-face culling.
- **refine_seamlines** (*bool*) – Refine seamlines based on image content.
- **projection** (Metashape.OrthoProjection) – Output projection.
- **region** (Metashape.BBox) – Region to be processed.
- **resolution** (*float*) – Pixel size in meters.
- **resolution_x** (*float*) – Pixel size in the X dimension in projected units.
- **resolution_y** (*float*) – Pixel size in the Y dimension in projected units.
- **flip_x** (*bool*) – Flip X axis direction.

- **flip_y** (*bool*) – Flip Y axis direction.
- **flip_z** (*bool*) – Flip Z axis direction.
- **subdivide_task** (*bool*) – Enable fine-level task subdivision.
- **workitem_size_cameras** (*int*) – Number of cameras in a workitem.
- **workitem_size_tiles** (*int*) – Number of tiles in a workitem.
- **max_workgroup_size** (*int*) – Maximum workgroup size.
- **replace_asset** (*bool*) – Replace default asset with generated orthomosaic.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

buildPanorama(*blending_mode=MosaicBlending*, *ghosting_filter=False*[, *rotation*][, *region*], *width=0*, *height=0*[, *camera_groups*][, *frames*][, *progress*])

Generate spherical panoramas from camera stations.

Parameters

- **blending_mode** (Metashape.BlendingMode) – Panorama blending mode.
- **ghosting_filter** (*bool*) – Enable ghosting filter.
- **rotation** (Metashape.Matrix) – Panorama 3x3 orientation matrix.
- **region** (Metashape.BBox) – Region to be generated.
- **width** (*int*) – Width of output panorama.
- **height** (*int*) – Height of output panorama.
- **camera_groups** (*list[int]*) – List of camera groups to process.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

buildPointCloud(*source_data=DepthMapsData*, *point_colors=True*, *point_confidence=False*, *keep_depth=True*, *max_neighbors=100*, *uniform_sampling=True*, *points_spacing=0.1*[, *asset*], *subdivide_task=True*, *workitem_size_cameras=20*, *max_workgroup_size=100*, *replace_asset=False*[, *frames*][, *progress*])

Generate point cloud for the chunk.

Parameters

- **source_data** (Metashape.DataSource) – Source data to extract points from.
- **point_colors** (*bool*) – Enable point colors calculation.
- **point_confidence** (*bool*) – Enable point confidence calculation.
- **keep_depth** (*bool*) – Enable store depth maps option.
- **max_neighbors** (*int*) – Maximum number of neighbor images to use for depth map filtering.
- **uniform_sampling** (*bool*) – Enable uniform point sampling.
- **points_spacing** (*float*) – Desired point spacing (m).
- **asset** (*int*) – Asset to process.
- **subdivide_task** (*bool*) – Enable fine-level task subdivision.

- **workitem_size_cameras** (*int*) – Number of cameras in a workitem.
- **max_workgroup_size** (*int*) – Maximum workgroup size.
- **replace_asset** (*bool*) – Replace default asset with generated point cloud.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

buildSeamlines(*epsilon=1.5*[, *progress*])

Generate shapes for orthomosaic seamlines.

Parameters

- **epsilon** (*float*) – Contour simplificaion threshold.
- **progress** (*Callable[[float], None]*) – Progress callback.

buildTexture(*blending_mode=MosaicBlending*, *texture_size=8192*, *fill_holes=True*, *ghosting_filter=True*[, *cameras*], *texture_type=DiffuseMap*[, *source_model*], *transfer_texture=True*, *workitem_size_cameras=20*, *max_workgroup_size=100*[, *progress*])

Generate texture for the chunk.

Parameters

- **blending_mode** (Metashape.BlendingMode) – Texture blending mode.
- **texture_size** (*int*) – Texture page size.
- **fill_holes** (*bool*) – Enable hole filling.
- **ghosting_filter** (*bool*) – Enable ghosting filter.
- **cameras** (*list[int]*) – A list of cameras to be used for texturing.
- **texture_type** (Metashape.Model.TextureType) – Texture type.
- **source_model** (*int*) – Source model.
- **transfer_texture** (*bool*) – Transfer texture.
- **workitem_size_cameras** (*int*) – Number of cameras in a workitem (block model only).
- **max_workgroup_size** (*int*) – Maximum workgroup size (block model only).
- **progress** (*Callable[[float], None]*) – Progress callback.

buildTiledModel(*pixel_size=0*, *tile_size=256*, *source_data=DepthMapsData*, *face_count=20000*, *ghosting_filter=False*, *transfer_texture=False*, *keep_depth=True*, *merge=False*[, *operand_chunk*][, *operand_frame*][, *operand_asset*][, *classes*], *subdivide_task=True*, *workitem_size_cameras=20*, *max_workgroup_size=100*, *replace_asset=False*[, *frames*][, *progress*])

Build tiled model for the chunk.

Parameters

- **pixel_size** (*float*) – Target model resolution in meters.
- **tile_size** (*int*) – Size of tiles in pixels.
- **source_data** (Metashape.DataSource) – Selects between point cloud and mesh.
- **face_count** (*int*) – Number of faces per megapixel of texture resolution.
- **ghosting_filter** (*bool*) – Enable ghosting filter.
- **transfer_texture** (*bool*) – Transfer source model texture to tiled model.

- **keep_depth** (*bool*) – Enable store depth maps option.
- **merge** (*bool*) – Merge tiled model flag.
- **operand_chunk** (*int*) – Operand chunk key.
- **operand_frame** (*int*) – Operand frame key.
- **operand_asset** (*int*) – Operand asset key.
- **classes** (*list[int]*) – List of point classes to be used for surface extraction.
- **subdivide_task** (*bool*) – Enable fine-level task subdivision.
- **workitem_size_cameras** (*int*) – Number of cameras in a workitem.
- **max_workgroup_size** (*int*) – Maximum workgroup size.
- **replace_asset** (*bool*) – Replace default asset with generated tiled model.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

buildUV(*mapping_mode=GenericMapping*, *page_count=1*, *texture_size=8192*, *pixel_size=0*[, *camera*][, *progress*])

Generate uv mapping for the model.

Parameters

- **mapping_mode** (Metashape.MappingMode) – Texture mapping mode.
- **page_count** (*int*) – Number of texture pages to generate.
- **texture_size** (*int*) – Expected size of texture page at texture generation step.
- **pixel_size** (*float*) – Texture resolution in meters.
- **camera** (*int*) – Camera to be used for texturing in CameraMapping mode.
- **progress** (*Callable[[float], None]*) – Progress callback.

calculatePointNormals(*point_neighbors=28*[, *point_cloud*][, *progress*])

Calculate point cloud normals.

Parameters

- **point_neighbors** (*int*) – Number of point neighbors to use for normal estimation.
- **point_cloud** (*int*) – Point cloud key to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

calibrateColors(*source_data=ModelData*, *white_balance=False*[, *cameras*][, *progress*])

Perform radiometric calibration.

Parameters

- **source_data** (Metashape.DataSource) – Source data for calibration.
- **white_balance** (*bool*) – Calibrate white balance.
- **cameras** (*list[int]*) – List of cameras to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

calibrateReflectance(*use_reflectance_panels=True*, *use_sun_sensor=False*[, *progress*])

Calibrate reflectance factors based on calibration panels and/or sun sensor.

Parameters

- **use_reflectance_panels** (*bool*) – Use calibrated reflectance panels.
- **use_sun_sensor** (*bool*) – Apply irradiance sensor measurements.
- **progress** (*Callable[[float], None]*) – Progress callback.

camera_crs

Coordinate system used for camera reference data.

Type
Metashape.CoordinateSystem

camera_groups

List of camera groups in the chunk.

Type
list[*Metashape.CameraGroup*]

camera_location_accuracy

Expected accuracy of camera coordinates in meters.

Type
Metashape.Vector

camera_rotation_accuracy

Expected accuracy of camera orientation angles in degrees.

Type
Metashape.Vector

camera_track

Camera track.

Type
Metashape.CameraTrack

camera_tracks

List of camera tracks in the chunk.

Type
list[*Metashape.CameraTrack*]

cameras

List of Regular and Keyframe cameras in the chunk.

Type
list[*Metashape.Camera*]

cir_transform

CIR calibration matrix.

Type
Metashape.CirTransform

colorizeModel(*source_data=ImagesData*[, *model*][, *progress*])

Calculate vertex colors for the model.

Parameters

- **source_data** (`Metashape.DataSource`) – Source data to extract colors from.
- **model** (*int*) – Key of model to colorize.
- **progress** (*Callable[[float], None]*) – Progress callback.

colorizePointCloud(*source_data=ImagesData*, *workitem_size_cameras=20*, *max_workgroup_size=100*, *subdivide_task=True*[, *point_cloud*][, *progress*])

Calculate point colors for the point cloud.

Parameters

- **source_data** (`Metashape.DataSource`) – Source data to extract colors from.
- **workitem_size_cameras** (*int*) – Number of cameras in a workitem.
- **max_workgroup_size** (*int*) – Maximum workgroup size.
- **subdivide_task** (*bool*) – Enable fine-level task subdivision.
- **point_cloud** (*int*) – Point cloud key to colorize.
- **progress** (*Callable[[float], None]*) – Progress callback.

component

Component.

Type
Metashape.Component

components

List of components in the chunk.

Type
list[*Metashape.Component*]

copy([*frames*][, *items*], *keypoints=True*[, *progress*])

Make a copy of the chunk.

Parameters

- **frames** (*list[*`Metashape.Chunk`*]*) – Optional list of frames to be copied.
- **items** (*list[*`Metashape.DataSource`*]*) – A list of items to copy.
- **keypoints** (*bool*) – copy key points data.
- **progress** (*Callable[[float], None]*) – Progress callback.

Returns
Copy of the chunk.

Return type
Metashape.Chunk

crs

Coordinate system used for reference data.

Type
Metashape.CoordinateSystem

decimateModel(*face_count=200000*[, *model*], *apply_to_selection=False*, *replace_asset=False*[, *frames*][, *progress*])

Decimate the model to the specified face count.

Parameters

- **face_count** (*int*) – Target face count.
- **model** (*int*) – Model to process.
- **apply_to_selection** (*bool*) – Apply to selection.
- **replace_asset** (*bool*) – Replace source model with decimated model.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

depth_maps

Default depth maps set for the current frame.

Type
Metashape.DepthMaps

depth_maps_sets

List of depth maps sets for the current frame.

Type
list[*Metashape.DepthMaps*]

detectFiducials(*generate_masks=False*, *mask_dark_pixels=True*, *generic_detector=True*, *right_angle_detector=False*, *v_shape_detector=False*, *frame_detector=False*, *fiducials_position_corners=True*, *fiducials_position_sides=True*[, *cameras*][, *frames*][, *progress*])

Detect fiducial marks on film cameras.

Parameters

- **generate_masks** (*bool*) – Generate background masks.
- **mask_dark_pixels** (*bool*) – Mask out dark pixels near frame edge.
- **generic_detector** (*bool*) – Use generic detector.
- **right_angle_detector** (*bool*) – Use right angle detector.
- **v_shape_detector** (*bool*) – Detect V-shape fiducials.
- **frame_detector** (*bool*) – Detect frame.
- **fiducials_position_corners** (*bool*) – Search corners for fiducials.
- **fiducials_position_sides** (*bool*) – Search sides for fiducials.
- **cameras** (*list[int]*) – List of cameras to process.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

detectMarkers(*target_type=CircularTarget12bit*, *tolerance=50*, *filter_mask=False*, *inverted=False*, *noparity=False*, *maximum_residual=5*, *minimum_size=0*, *minimum_dist=5*[, *cameras*][, *frames*][, *progress*])

Create markers from coded targets.

Parameters

- **target_type** (Metashape.TargetType) – Type of targets.
- **tolerance** (*int*) – Detector tolerance (0 - 100).

- **filter_mask** (*bool*) – Ignore masked image regions.
- **inverted** (*bool*) – Detect markers on black background.
- **noparity** (*bool*) – Disable parity checking.
- **maximum_residual** (*float*) – Maximum residual for non-coded targets in pixels.
- **minimum_size** (*int*) – Minimum target radius in pixels to be detected (CrossTarget type only).
- **minimum_dist** (*int*) – Minimum distance between targets in pixels (CrossTarget type only).
- **cameras** (*list[int]*) – List of cameras to process.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

detectPowerlines(*min_altitude=1*, *n_points_per_line=100*, *max_quantization_error=0.01*, *use_model=True*[, *progress*])

Detect powerlines for the chunk.

Parameters

- **min_altitude** (*float*) – Minimum altitude for reconstructed powerlines.
- **n_points_per_line** (*int*) – Maximum number of vertices per detected line.
- **max_quantization_error** (*float*) – Maximum allowed distance between polyline and smooth continuous curve.
- **use_model** (*bool*) – Use model for visibility checks.
- **progress** (*Callable[[float], None]*) – Progress callback.

elevation

Default elevation model for the current frame.

Type
Metashape.Elevation

elevations

List of elevation models for the current frame.

Type
list[*Metashape.Elevation*]

enabled

Enables/disables the chunk.

Type
bool

euler_angles

Euler angles triplet used for rotation reference.

Type
Metashape.EulerAngles

exportCameras(*path='', format=CamerasFormatXML*[, *crs*], *save_points=True, save_markers=False, save_invalid_matches=False, use_labels=False, use_initial_calibration=False, image_orientation=0, chan_rotation_order=RotationOrderXYZ, binary=False, bundler_save_list=True, bundler_path_list='list.txt', bingo_save_image=True, bingo_save_itera=True, bingo_save_geoin=True, bingo_save_gps=False, bingo_path_itera='itera.dat', bingo_path_image='image.dat', bingo_path_geoin='geoin.dat', bingo_path_gps='gps-imu.dat'*[, *progress*])

Export point cloud and/or camera positions.

Parameters

- **path** (*str*) – Path to output file.
- **format** (Metashape.CamerasFormat) – Export format.
- **crs** (Metashape.CoordinateSystem) – Output coordinate system.
- **save_points** (*bool*) – Enables/disables export of automatic tie points.
- **save_markers** (*bool*) – Enables/disables export of manual matching points.
- **save_invalid_matches** (*bool*) – Enables/disables export of invalid image matches.
- **use_labels** (*bool*) – Enables/disables label based item identifiers.
- **use_initial_calibration** (*bool*) – Transform image coordinates to initial calibration.
- **image_orientation** (*int*) – Image coordinate system (0 - X right, 1 - X up, 2 - X left, 3 - X down).
- **chan_rotation_order** (Metashape.RotationOrder) – Rotation order (CHAN format only).
- **binary** (*bool*) – Enables/disables binary encoding for selected format (if applicable).
- **bundler_save_list** (*bool*) – Enables/disables export of Bundler image list file.
- **bundler_path_list** (*str*) – Path to Bundler image list file.
- **bingo_save_image** (*bool*) – Enables/disables export of BINGO IMAGE COORDINATE file.
- **bingo_save_itera** (*bool*) – Enables/disables export of BINGO ITERA file.
- **bingo_save_geoin** (*bool*) – Enables/disables export of BINGO GEO INPUT file.
- **bingo_save_gps** (*bool*) – Enables/disables export of BINGO GPS/IMU data.
- **bingo_path_itera** (*str*) – Path to BINGO ITERA file.
- **bingo_path_image** (*str*) – Path to BINGO IMAGE COORDINATE file.
- **bingo_path_geoin** (*str*) – Path to BINGO GEO INPUT file.
- **bingo_path_gps** (*str*) – Path to BINGO GPS/IMU file.
- **progress** (*Callable[[float], None]*) – Progress callback.

exportMarkers(*path=''*[, *crs*], *binary=False*[, *progress*])

Export markers.

Parameters

- **path** (*str*) – Path to output file.
- **crs** (Metashape.CoordinateSystem) – Output coordinate system.

- **binary** (*bool*) – Enables/disables binary encoding for selected format (if applicable).
- **progress** (*Callable[[float], None]*) – Progress callback.

exportModel(*path=''*, *binary=True*, *precision=6*, *texture_format=ImageFormatJPEG*, *save_texture=True*, *save_uv=True*, *save_normals=True*, *save_colors=True*, *save_confidence=False*, *save_cameras=True*, *save_markers=True*, *save_udim=False*, *save_alpha=False*, *embed_texture=False*, *strip_extensions=False*, *raster_transform=RasterTransformNone*, *colors_rgb_8bit=True*, *comment=''*, *save_comment=True*, *format=ModelFormatNone*[, *crs*][, *shift*], *clip_to_boundary=True*, *save_metadata_xml=False*[, *model*][, *viewpoint*][, *progress*])

Export generated model for the chunk.

Parameters

- **path** (*str*) – Path to output model.
- **binary** (*bool*) – Enables/disables binary encoding (if supported by format).
- **precision** (*int*) – Number of digits after the decimal point (for text formats).
- **texture_format** (Metashape.ImageFormat) – Texture format.
- **save_texture** (*bool*) – Enables/disables texture export.
- **save_uv** (*bool*) – Enables/disables uv coordinates export.
- **save_normals** (*bool*) – Enables/disables export of vertex normals.
- **save_colors** (*bool*) – Enables/disables export of vertex colors.
- **save_confidence** (*bool*) – Enables/disables export of vertex confidence.
- **save_cameras** (*bool*) – Enables/disables camera export.
- **save_markers** (*bool*) – Enables/disables marker export.
- **save_udim** (*bool*) – Enables/disables UDIM texture layout.
- **save_alpha** (*bool*) – Enables/disables alpha channel export.
- **embed_texture** (*bool*) – Embeds texture inside the model file (if supported by format).
- **strip_extensions** (*bool*) – Strips camera label extensions during export.
- **raster_transform** (Metashape.RasterTransformType) – Raster band transformation.
- **colors_rgb_8bit** (*bool*) – Convert colors to 8 bit RGB.
- **comment** (*str*) – Optional comment (if supported by selected format).
- **save_comment** (*bool*) – Enables/disables comment export.
- **format** (Metashape.ModelFormat) – Export format.
- **crs** (Metashape.CoordinateSystem) – Output coordinate system.
- **shift** (Metashape.Vector) – Optional shift to be applied to vertex coordinates.
- **clip_to_boundary** (*bool*) – Clip model to boundary shapes.
- **save_metadata_xml** (*bool*) – Save metadata.xml file.
- **model** (*int*) – Model key to export.
- **viewpoint** (Metashape.Viewpoint) – Default view.
- **progress** (*Callable[[float], None]*) – Progress callback.

exportOrthophotos(*path='{filename}.tif'* [, *cameras*], *raster_transform=RasterTransformNone* [, *projection*][, *region*], *resolution=0*, *resolution_x=0*, *resolution_y=0*, *save_kml=False*, *save_world=False*, *save_alpha=True* [, *image_compression*], *white_background=True*, *north_up=True* [, *progress*])

Export orthophotos for the chunk.

Parameters

- **path** (*str*) – Path to output orthophoto.
- **cameras** (*list[int]*) – List of cameras to process.
- **raster_transform** (Metashape.RasterTransformType) – Raster band transformation.
- **projection** (Metashape.OrthoProjection) – Output projection.
- **region** (Metashape.BBox) – Region to be exported.
- **resolution** (*float*) – Output resolution in meters.
- **resolution_x** (*float*) – Pixel size in the X dimension in projected units.
- **resolution_y** (*float*) – Pixel size in the Y dimension in projected units.
- **save_kml** (*bool*) – Enable kml file generation.
- **save_world** (*bool*) – Enable world file generation.
- **save_alpha** (*bool*) – Enable alpha channel generation.
- **image_compression** (Metashape.ImageCompression) – Image compression parameters.
- **white_background** (*bool*) – Enable white background.
- **north_up** (*bool*) – Use north-up orientation for export.
- **progress** (*Callable[[float], None]*) – Progress callback.

exportPointCloud(*path=''*, *source_data=PointCloudData* [, *point_cloud*], *binary=True*, *save_point_color=True*, *save_point_normal=True*, *save_point_intensity=True*, *save_point_classification=True*, *save_point_confidence=True*, *save_point_return_number=True*, *save_point_scan_angle=True*, *save_point_source_id=True*, *save_point_timestamp=True*, *save_point_index=True*, *raster_transform=RasterTransformNone*, *colors_rgb_8bit=True*, *comment=''*, *save_comment=True*, *format=PointCloudFormatNone*, *image_format=ImageFormatJPEG* [, *crs*][, *shift*][, *region*], *clip_to_boundary=True*, *block_width=1000*, *block_height=1000*, *split_in_blocks=False* [, *classes*], *save_images=False*, *compression=True*, *tileset_version='1.0'*, *screen_space_error=16*, *folder_depth=5* [, *viewpoint*], *subdivide_task=True* [, *progress*])

Export point cloud.

Parameters

- **path** (*str*) – Path to output file.
- **source_data** (Metashape.DataSource) – Selects between point cloud and tie points. If not specified, uses point cloud if available.
- **point_cloud** (*int*) – Point cloud key to export.
- **binary** (*bool*) – Enables/disables binary encoding for selected format (if applicable).
- **save_point_color** (*bool*) – Enables/disables export of point color.

- **save_point_normal** (*bool*) – Enables/disables export of point normal.
- **save_point_intensity** (*bool*) – Enables/disables export of point intensity.
- **save_point_classification** (*bool*) – Enables/disables export of point classification.
- **save_point_confidence** (*bool*) – Enables/disables export of point confidence.
- **save_point_return_number** (*bool*) – Enables/disables export of point return number.
- **save_point_scan_angle** (*bool*) – Enables/disables export of point scan angle.
- **save_point_source_id** (*bool*) – Enables/disables export of point source ID.
- **save_point_timestamp** (*bool*) – Enables/disables export of point timestamp.
- **save_point_index** (*bool*) – Enables/disables export of point row and column indices.
- **raster_transform** (`Metashape.RasterTransformType`) – Raster band transformation.
- **colors_rgb_8bit** (*bool*) – Convert colors to 8 bit RGB.
- **comment** (*str*) – Optional comment (if supported by selected format).
- **save_comment** (*bool*) – Enable comment export.
- **format** (`Metashape.PointCloudFormat`) – Export format.
- **image_format** (`Metashape.ImageFormat`) – Image data format.
- **crs** (`Metashape.CoordinateSystem`) – Output coordinate system.
- **shift** (`Metashape.Vector`) – Optional shift to be applied to point coordinates.
- **region** (`Metashape.BBox`) – Region to be exported.
- **clip_to_boundary** (*bool*) – Clip point cloud to boundary shapes.
- **block_width** (*float*) – Block width in meters.
- **block_height** (*float*) – Block height in meters.
- **split_in_blocks** (*bool*) – Enable tiled export.
- **classes** (*list[int]*) – List of point classes to be exported.
- **save_images** (*bool*) – Enable image export.
- **compression** (*bool*) – Enable compression (Cesium format only).
- **tileset_version** (*str*) – Cesium 3D Tiles format version to export (1.0 or 1.1).
- **screen_space_error** (*float*) – Target screen space error (Cesium format only).
- **folder_depth** (*int*) – Tileset subdivision depth (Cesium format only).
- **viewpoint** (`Metashape.Viewpoint`) – Default view.
- **subdivide_task** (*bool*) – Enable fine-level task subdivision.
- **progress** (*Callable[[float], None]*) – Progress callback.

exportRaster(*path='', format=RasterFormatTiles, image_format=ImageFormatNone, raster_transform=RasterTransformNone*[, *projection*][, *region*], *resolution=0, resolution_x=0, resolution_y=0, block_width=10000, block_height=10000, split_in_blocks=False, width=0, height=0*[, *world_transform*], *nodata_value=-32767, save_kml=False, save_world=False, save_scheme=False, save_alpha=True, image_description=''*[, *image_compression*], *network_links=True, global_profile=False, min_zoom_level=-1, max_zoom_level=-1, white_background=True, clip_to_boundary=True, title='Orthomosaic', description='Generated by Agisoft Metashape', source_data=OrthomosaicData*[, *asset*], *north_up=True, tile_width=256, tile_height=256*[, *progress*])

Export DEM or orthomosaic to file.

Parameters

- **path** (*str*) – Path to output orthomosaic.
- **format** (Metashape.RasterFormat) – Export format.
- **image_format** (Metashape.ImageFormat) – Tile format.
- **raster_transform** (Metashape.RasterTransformType) – Raster band transformation.
- **projection** (Metashape.OrthoProjection) – Output projection.
- **region** (Metashape.BBox) – Region to be exported.
- **resolution** (*float*) – Output resolution in meters.
- **resolution_x** (*float*) – Pixel size in the X dimension in projected units.
- **resolution_y** (*float*) – Pixel size in the Y dimension in projected units.
- **block_width** (*int*) – Raster block width in pixels.
- **block_height** (*int*) – Raster block height in pixels.
- **split_in_blocks** (*bool*) – Split raster in blocks.
- **width** (*int*) – Raster width.
- **height** (*int*) – Raster height.
- **world_transform** (Metashape.Matrix) – 2x3 raster-to-world transformation matrix.
- **nodata_value** (*float*) – No-data value (DEM export only).
- **save_kml** (*bool*) – Enable kml file generation.
- **save_world** (*bool*) – Enable world file generation.
- **save_scheme** (*bool*) – Enable tile scheme files generation.
- **save_alpha** (*bool*) – Enable alpha channel generation.
- **image_description** (*str*) – Optional description to be added to image files.
- **image_compression** (Metashape.ImageCompression) – Image compression parameters.
- **network_links** (*bool*) – Enable network links generation for KMZ format.
- **global_profile** (*bool*) – Use global profile (GeoPackage format only).
- **min_zoom_level** (*int*) – Minimum zoom level (GeoPackage, Google Map Tiles, MBTiles and World Wind Tiles formats only).

- **max_zoom_level** (*int*) – Maximum zoom level (GeoPackage, Google Map Tiles, MBTiles and World Wind Tiles formats only).
- **white_background** (*bool*) – Enable white background.
- **clip_to_boundary** (*bool*) – Clip raster to boundary shapes.
- **title** (*str*) – Export title.
- **description** (*str*) – Export description.
- **source_data** (Metashape.DataSource) – Selects between DEM and orthomosaic.
- **asset** (*int*) – Asset key to export.
- **north_up** (*bool*) – Use north-up orientation for export.
- **tile_width** (*int*) – Tile width in pixels.
- **tile_height** (*int*) – Tile height in pixels.
- **progress** (*Callable[[float], None]*) – Progress callback.

exportReference(*path=''*, *format=ReferenceFormatNone*, *items=ReferenceItemsCameras*, *columns=''*, *delimiter=' '*, *precision=6*[, *progress*])

Export reference data to the specified file.

Parameters

- **path** (*str*) – Path to the output file.
- **format** (Metashape.ReferenceFormat) – Export format.
- **items** (Metashape.ReferenceItems) – Items to export in CSV format.
- **columns** (*str*) – Column order in csv format (n - label, o - enabled flag, x/y/z - coordinates, X/Y/Z - coordinate accuracy, a/b/c - rotation angles, A/B/C - rotation angle accuracy, u/v/w - estimated coordinates, U/V/W - coordinate errors, d/e/f - estimated orientation angles, D/E/F - orientation errors, p/q/r - estimated coordinates variance, i/j/k - estimated orientation angles variance, [] - group of multiple values, | - column separator within group).
- **delimiter** (*str*) – Column delimiter in csv format.
- **precision** (*int*) – Number of digits after the decimal point (for CSV format).
- **progress** (*Callable[[float], None]*) – Progress callback.

exportReport(*path=''*, *title=''*, *description=''*, *font_size=12*, *page_numbers=True*, *include_system_info=True*[, *user_settings*][, *progress*])

Export processing report in PDF format.

Parameters

- **path** (*str*) – Path to output report.
- **title** (*str*) – Report title.
- **description** (*str*) – Report description.
- **font_size** (*int*) – Font size (pt).
- **page_numbers** (*bool*) – Enable page numbers.
- **include_system_info** (*bool*) – Include system information.
- **user_settings** (*list[tuple[str, str]]*) – A list of user defined settings to include on the Processing Parameters page.

- **progress** (*Callable[[float], None]*) – Progress callback.

exportShapes(*path='', save_points=False, save_polylines=False, save_polygons=False*[, *groups*], *format=ShapesFormatNone*[, *crs*][, *shift*], *polygons_as_polylines=False*, *save_labels=True*, *save_attributes=True*[, *progress*])

Export shapes layer to file.

Parameters

- **path** (*str*) – Path to shape file.
- **save_points** (*bool*) – Export points.
- **save_polylines** (*bool*) – Export polylines.
- **save_polygons** (*bool*) – Export polygons.
- **groups** (*list[int]*) – A list of shape groups to export.
- **format** (Metashape.ShapesFormat) – Export format.
- **crs** (Metashape.CoordinateSystem) – Output coordinate system.
- **shift** (Metashape.Vector) – Optional shift to be applied to vertex coordinates.
- **polygons_as_polylines** (*bool*) – Save polygons as polylines.
- **save_labels** (*bool*) – Export labels.
- **save_attributes** (*bool*) – Export attributes.
- **progress** (*Callable[[float], None]*) – Progress callback.

exportTexture(*path='', texture_type=DiffuseMap, raster_transform=RasterTransformNone, save_alpha=False*[, *progress*])

Export model texture to file.

Parameters

- **path** (*str*) – Path to output file.
- **texture_type** (Metashape.Model.TextureType) – Texture type.
- **raster_transform** (Metashape.RasterTransformType) – Raster band transformation.
- **save_alpha** (*bool*) – Enable alpha channel export.
- **progress** (*Callable[[float], None]*) – Progress callback.

exportTiledModel(*path='', format=TiledModelFormatNone, model_format=ModelFormatCOLLADA, texture_format=ImageFormatJPEG, raster_transform=RasterTransformNone*[, *image_compression*][, *crs*], *clip_to_boundary=True*[, *tiled_model*], *model_compression=True, tileset_version='1.0', use_tileset_transform=True, screen_space_error=16, folder_depth=5*[, *model_group*], *pixel_size=0, tile_size=256, face_count=20000*[, *progress*])

Export generated tiled model for the chunk.

Parameters

- **path** (*str*) – Path to output model.
- **format** (Metashape.TiledModelFormat) – Export format.
- **model_format** (Metashape.ModelFormat) – Model format for zip export.

- **texture_format** (Metashape.ImageFormat) – Texture format.
- **raster_transform** (Metashape.RasterTransformType) – Raster band transformation.
- **image_compression** (Metashape.ImageCompression) – Image compression parameters.
- **crs** (Metashape.CoordinateSystem) – Output coordinate system.
- **clip_to_boundary** (*bool*) – Clip tiled model to boundary shapes.
- **tiled_model** (*int*) – Tiled model key to export.
- **model_compression** (*bool*) – Enable mesh compression (Cesium format only).
- **tileset_version** (*str*) – Cesium 3D Tiles format version to export (1.0 or 1.1).
- **use_tileset_transform** (*bool*) – Use tileset transform instead of individual tile transforms (Cesium format only).
- **screen_space_error** (*float*) – Target screen space error (Cesium format only).
- **folder_depth** (*int*) – Tileset subdivision depth (Cesium format only).
- **model_group** (*int*) – Block model key to export.
- **pixel_size** (*float*) – Target model resolution in meters (block model export only).
- **tile_size** (*int*) – Size of tiles in pixels (block model export only).
- **face_count** (*int*) – Number of faces per megapixel of texture resolution (block model export only).
- **progress** (*Callable[[float], None]*) – Progress callback.

filterPointCloud(*point_spacing=0*[, *point_cloud*], *replace_asset=False*[, *frames*][, *progress*])

Reduce point cloud points number.

Parameters

- **point_spacing** (*float*) – Desired point spacing (m).
- **point_cloud** (*int*) – Point cloud key to filter.
- **replace_asset** (*bool*) – Replace default asset with filtered point cloud.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

findCamera(*key*)

Find camera by its key.

Returns
Found camera.

Return type
Metashape.Camera

findCameraGroup(*key*)

Find camera group by its key.

Returns
Found camera group.

Return type
Metashape.CameraGroup

findCameraTrack(*key*)

Find camera track by its key.

Returns
Found camera track.

Return type
Metashape.CameraTrack

findDepthMaps(*key*)

Find depth maps by its key.

Returns
Found depth maps.

Return type
Metashape.DepthMaps

findElevation(*key*)

Find elevation model by its key.

Returns
Found elevation model.

Return type
Metashape.Elevation

findFrame(*key*)

Find frame by its key.

Returns
Found frame.

Return type
Metashape.Chunk

findMarker(*key*)

Find marker by its key.

Returns
Found marker.

Return type
Metashape.Marker

findMarkerGroup(*key*)

Find marker group by its key.

Returns
Found marker group.

Return type
Metashape.MarkerGroup

findModel(*key*)

Find model by its key.

Returns
Found model.

Return type
Metashape.Model

findOrthomosaic(*key*)

Find orthomosaic by its key.

Returns
Found orthomosaic.

Return type
Metashape.Orthomosaic

findPointCloud(*key*)

Find point cloud by its key.

Returns
Found point cloud.

Return type
Metashape.PointCloud

findPointCloudGroup(*key*)

Find point cloud group by its key.

Parameters
key (*int*) – Point cloud group key.

Returns
Found point cloud group.

Return type
Metashape.PointCloudGroup

findScalebar(*key*)

Find scalebar by its key.

Returns
Found scalebar.

Return type
Metashape.Scalebar

findScalebarGroup(*key*)

Find scalebar group by its key.

Returns
Found scalebar group.

Return type
Metashape.ScalebarGroup

findSensor(*key*)

Find sensor by its key.

Returns
Found sensor.

Return type
Metashape.Sensor

findTiledModel(*key*)

Find tiled model by its key.

Returns
Found tiled model.

Return type
Metashape.TiledModel

frame

Current frame index.

Type
int

frames

List of frames in the chunk.

Type
list[*Metashape.Chunk*]

generateMasks(*path='{filename}_mask.png'*, *masking_mode=MaskingModeAlpha*, *mask_operation=MaskOperationReplacement*, *tolerance=10*[, *cameras*], *mask_defocus=False*, *fix_coverage=True*, *blur_threshold=3*, *depth_threshold=3.40282e+38*[, *progress*])

Generate masks for multiple cameras.

Parameters

- **path** (`str`) – Mask file name template.
- **masking_mode** (`Metashape.MaskingMode`) – Mask generation mode.
- **mask_operation** (`Metashape.MaskOperation`) – Mask operation.
- **tolerance** (`int`) – Background masking tolerance.
- **cameras** (`list[int]`) – Optional list of cameras to be processed.
- **mask_defocus** (`bool`) – Mask defocus areas.
- **fix_coverage** (`bool`) – Extend masks to cover whole mesh (only if mask_defocus=True).
- **blur_threshold** (`float`) – Allowed blur radius on a photo in pix (only if mask_defocus=True).
- **depth_threshold** (`float`) – Maximum depth of masked areas in meters (only if mask_defocus=False).
- **progress** (`Callable[[float], None]`) – Progress callback.

generatePrescriptionMap(*class_count=4*, *cell_size=1*, *classification_method=JenksNaturalBreaksClassification*[, *boundary_shape_group*][, *breakpoints*][, *rates*][, *progress*])

Generate prescription map for orthomosaic.

Parameters

- **class_count** (`int`) – Number of classes.
- **cell_size** (`float`) – Step of prescription grid, meters.
- **classification_method** (`Metashape.ClassificationMethod`) – Index values classification method.

- **boundary_shape_group** (*int*) – Boundary shape group.
- **breakpoints** (*list[float]*) – Classification breakpoints.
- **rates** (*list[float]*) – Fertilizer rate for each class.
- **progress** (*Callable[[float], None]*) – Progress callback.

image_brightness

Image brightness as percentage.

Type

float

image_contrast

Image contrast as percentage.

Type

float

importCameras(*path='', format=CamerasFormatXML*[, *crs*], *image_orientation=0*, *image_list='list.txt'*, *load_image_list=False*[, *progress*])

Import camera positions.

Parameters

- **path** (*str*) – Path to the file.
- **format** (Metashape.CamerasFormat) – File format.
- **crs** (Metashape.CoordinateSystem) – Ground coordinate system.
- **image_orientation** (*int*) – Image coordinate system (0 - X right, 1 - X up, 2 - X left, 3 - X down).
- **image_list** (*str*) – Path to image list file (Bundler format only).
- **load_image_list** (*bool*) – Enable Bundler image list import.
- **progress** (*Callable[[float], None]*) – Progress callback.

importDepthImages(*format=PointCloudFormatNone*[, *filenames*][, *color_filenames*], *image_path=''*, *multiplane=False*[, *progress*])

Import images with depth data.

Parameters

- **format** (Metashape.PointCloudFormat) – Point cloud format.
- **filenames** (*list[str]*) – List of files to import.
- **color_filenames** (*list[str]*) – List of corresponding color files, if present.
- **image_path** (*str*) – Path template to output files.
- **multiplane** (*bool*) – Import as a multi-camera system
- **progress** (*Callable[[float], None]*) – Progress callback.

importMarkers(*path=''*[, *progress*])

Import markers.

Parameters

- **path** (*str*) – Path to the file.
- **progress** (*Callable[[float], None]*) – Progress callback.

importModel(*path=''*, *format=ModelFormatNone*[, *crs*][, *shift*], *decode_udim=True*, *replace_asset=False*[, *frame_paths*][, *progress*])

Import model from file.

Parameters

- **path** (*str*) – Path to model.
- **format** (Metashape.ModelFormat) – Model format.
- **crs** (Metashape.CoordinateSystem) – Model coordinate system.
- **shift** (Metashape.Vector) – Optional shift to be applied to vertex coordinates.
- **decode_udim** (*bool*) – Load UDIM texture layout.
- **replace_asset** (*bool*) – Replace default asset with imported model.
- **frame_paths** (*list[str]*) – List of model paths to import in each frame of a multiframe chunk.
- **progress** (*Callable[[float], None]*) – Progress callback.

importPointCloud(*path=''*, *format=PointCloudFormatNone*[, *crs*][, *shift*], *precision=0*, *is_laser_scan=False*, *replace_asset=False*, *import_images=True*, *calculate_normals=True*, *ignore_normals=False*, *point_neighbors=28*, *scanner_at_origin=False*, *ignore_scanner_origin=False*, *ignore_trajectory=False*[, *trajectory*][, *frame_paths*][, *progress*])

Import point cloud from file.

Parameters

- **path** (*str*) – Path to point cloud.
- **format** (Metashape.PointCloudFormat) – Point cloud format.
- **crs** (Metashape.CoordinateSystem) – Point cloud coordinate system.
- **shift** (Metashape.Vector) – Optional shift to be applied to point coordinates.
- **precision** (*float*) – Coordinate precision (m). For default precision use 0.
- **is_laser_scan** (*bool*) – Import point clouds as laser scans.
- **replace_asset** (*bool*) – Replace default asset with imported point cloud.
- **import_images** (*bool*) – Import images embedded in laser scan.
- **calculate_normals** (*bool*) – Calculate point normals.
- **ignore_normals** (*bool*) – Ignore normals in imported file.
- **point_neighbors** (*int*) – Number of point neighbors to use for normal estimation.
- **scanner_at_origin** (*bool*) – Use laser scan origin as scanner position for unstructured point clouds.
- **ignore_scanner_origin** (*bool*) – Do not use laser scan origin as scanner position for structured point clouds.
- **ignore_trajectory** (*bool*) – Do not attach trajectory to imported point cloud.
- **trajectory** (*int*) – Trajectory key to attach.
- **frame_paths** (*list[str]*) – List of point cloud paths to import in each frame of a multiframe chunk.

- **progress** (*Callable[[float], None]*) – Progress callback.

importRaster(*path=''*[, *crs*], *raster_type=ElevationData*, *nodata_value=-32767*, *has_nodata_value=False*, *replace_asset=False*[, *frames*][, *progress*])

Import DEM or orthomosaic from file.

Parameters

- **path** (*str*) – Path to elevation model in GeoTIFF format.
- **crs** (Metashape.CoordinateSystem) – Default coordinate system if not specified in GeoTIFF file.
- **raster_type** (Metashape.DataSource) – Type of raster layer to import.
- **nodata_value** (*float*) – No-data value.
- **has_nodata_value** (*bool*) – No-data value valid flag.
- **replace_asset** (*bool*) – Replace default raster with imported one.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

importReference(*path=''*, *format=ReferenceFormatCSV*, *columns=''*, *delimiter=''*, *group_delimiters=False*, *skip_rows=0*[, *items*][, *crs*], *ignore_labels=False*, *create_markers=False*, *threshold=0.1*, *shutter_lag=0*[, *progress*])

Import reference data from the specified file.

Parameters

- **path** (*str*) – Path to the file with reference data.
- **format** (Metashape.ReferenceFormat) – File format.
- **columns** (*str*) – Column order in csv format (n - label, o - enabled flag, x/y/z - coordinates, X/Y/Z - coordinate accuracy, a/b/c - rotation angles, A/B/C - rotation angle accuracy, [] - group of multiple values, | - column separator within group).
- **delimiter** (*str*) – Column delimiter in csv format.
- **group_delimiters** (*bool*) – Combine consecutive delimiters in csv format.
- **skip_rows** (*int*) – Number of rows to skip in (csv format only).
- **items** (Metashape.ReferenceItems) – List of items to load reference for (csv format only).
- **crs** (Metashape.CoordinateSystem) – Reference data coordinate system (csv format only).
- **ignore_labels** (*bool*) – Matches reference data based on coordinates alone (csv format only).
- **create_markers** (*bool*) – Create markers for missing entries (csv format only).
- **threshold** (*float*) – Error threshold in meters used when ignore_labels is set (csv format only).
- **shutter_lag** (*float*) – Shutter lag in seconds (APM format only).
- **progress** (*Callable[[float], None]*) – Progress callback.

importShapes(*path=''*, *replace=False*, *boundary_type=NoBoundary*, *format=ShapesFormatNone*, *columns='nxyzd'*, *delimiter=', '*, *group_delimiters=False*, *skip_rows=0*[, *crs*][, *progress*])

Import shapes layer from file.

Parameters

- **path** (*str*) – Path to shape file.
- **replace** (*bool*) – Replace current shapes with new data.
- **boundary_type** (Metashape.Shape.BoundaryType) – Boundary type to be applied to imported shapes.
- **format** (Metashape.ShapesFormat) – Shapes format.
- **columns** (*str*) – Column order in csv format (n - label, x/y/z - coordinates, d - description, [] - group of multiple values, | - column separator within group).
- **delimiter** (*str*) – Column delimiter in csv format.
- **group_delimiters** (*bool*) – Combine consequitive delimiters in csv format.
- **skip_rows** (*int*) – Number of rows to skip in (csv format only).
- **crs** (Metashape.CoordinateSystem) – Reference data coordinate system (csv format only).
- **progress** (*Callable[[float], None]*) – Progress callback.

importTiledModel(*path=''*[, *progress*])

Import tiled model from file.

Parameters

- **path** (*str*) – Path to tiled model.
- **progress** (*Callable[[float], None]*) – Progress callback.

importTrajectory(*path=''*, *format=TrajectoryFormatNone*, *columns='txyz'*, *delimiter=' '*, *skip_rows=0*[, *crs*][, *shift*], *replace_asset=False*[, *progress*])

Import trajectory from file.

Parameters

- **path** (*str*) – Trajectory file path.
- **format** (Metashape.TrajectoryFormat) – Trajectory format.
- **columns** (*str*) – Column order (t - time, x/y/z - coordinates, a/b/c - rotation angles, space - skip column).
- **delimiter** (*str*) – CSV delimiter.
- **skip_rows** (*int*) – Number of rows to skip.
- **crs** (Metashape.CoordinateSystem) – Point cloud coordinate system.
- **shift** (Metashape.Vector) – Optional shift to be applied to point coordinates.
- **replace_asset** (*bool*) – Replace default asset with imported trajectory.
- **progress** (*Callable[[float], None]*) – Progress callback.

importVideo(*path*, *image_path*, *frame_step=CustomFrameStep*, *custom_frame_step=1*, *time_start=0*, *time_end=-1*)

Imports video to active chunk.

Parameters

- **path** (*str*) – Path to source video.
- **image_path** (*str*) – Path to directory where to save frames with filename template. For example: /path/to/dir/frame{filenum}.png.
- **frame_step** (Metashape.FrameStep) – Frame step type.
- **custom_frame_step** (*int*) – Every custom_frame_step'th frame will be saved. Used for frame_step=CustomFrameStep.
- **time_start** (*int*) – The starting point for importing video, in milliseconds.
- **time_end** (*int*) – The endpoint for importing video, in milliseconds.

key

Chunk identifier.

Type
int

label

Chunk label.

Type
str

loadReferenceExif(*load_rotation=False*, *load_accuracy=False*)

Import camera locations from EXIF meta data.

Parameters

- **load_rotation** (*bool*) – load yaw, pitch and roll orientation angles.
- **load_accuracy** (*bool*) – load camera location accuracy.

loadReflectancePanelCalibration(*path*[, *cameras*])

Load reflectance panel calibration from CSV file.

Parameters

- **path** (*str*) – Path to calibration file.
- **cameras** (*list[*Metashape.Camera*]*) – List of cameras to process.

locateReflectancePanels([*progress*])

Locate reflectance panels based on QR-codes.

Parameters
progress (*Callable[[float], None]*) – Progress callback.

marker_crs

Coordinate system used for marker reference data.

Type
Metashape.CoordinateSystem

marker_groups

List of marker groups in the chunk.

Type
list[*Metashape.MarkerGroup*]

marker_location_accuracy

Expected accuracy of marker coordinates in meters.

Type

Metashape.Vector

marker_projection_accuracy

Expected accuracy of marker projections in pixels.

Type

float

markers

List of Regular, Vertex and Fiducial markers in the chunk.

Type

list[*Metashape.Marker*]

masks

Image masks.

Type

Metashape.Masks

matchPhotos(*downscale=1*, *downscale_3d=1*, *generic_preselection=True*, *reference_preselection=True*, *reference_preselection_mode=ReferencePreselectionSource*, *filter_mask=False*, *mask_tiepoints=True*, *filter_stationary_points=True*, *keypoint_limit=40000*, *keypoint_limit_3d=100000*, *keypoint_limit_per_mpx=1000*, *tiepoint_limit=4000*, *keep_keypoints=False*[, *pairs*][, *cameras*], *guided_matching=False*, *reset_matches=False*, *subdivide_task=True*, *workitem_size_cameras=20*, *workitem_size_pairs=80*, *max_workgroup_size=100*, *laser_scans_vertical_axis=0*, *match_laser_scans=False*[, *progress*])

Perform image matching for the chunk frame.

Parameters

- **downscale** (*int*) – Image alignment accuracy (0 - Highest, 1 - High, 2 - Medium, 4 - Low, 8 - Lowest).
- **downscale_3d** (*int*) – Laser scan alignment accuracy (1 - Highest, 2 - High, 4 - Medium, 8 - Low, 16 - Lowest).
- **generic_preselection** (*bool*) – Enable generic preselection.
- **reference_preselection** (*bool*) – Enable reference preselection.
- **reference_preselection_mode** (`Metashape.ReferencePreselectionMode`) – Reference preselection mode.
- **filter_mask** (*bool*) – Filter points by mask.
- **mask_tiepoints** (*bool*) – Apply mask filter to tie points.
- **filter_stationary_points** (*bool*) – Exclude tie points which are stationary across images.
- **keypoint_limit** (*int*) – Key point limit.
- **keypoint_limit_3d** (*int*) – Key point limit for laser scans.
- **keypoint_limit_per_mpx** (*int*) – Key point limit per megapixel.
- **tiepoint_limit** (*int*) – Tie point limit.

- **keep_keypoints** (*bool*) – Store keypoints in the project.
- **pairs** (*list[tuple[int, int]]*) – User defined list of camera pairs to match.
- **cameras** (*list[int]*) – List of cameras to match.
- **guided_matching** (*bool*) – Enable guided image matching.
- **reset_matches** (*bool*) – Reset current matches.
- **subdivide_task** (*bool*) – Enable fine-level task subdivision.
- **workitem_size_cameras** (*int*) – Number of cameras in a workitem.
- **workitem_size_pairs** (*int*) – Number of image pairs in a workitem.
- **max_workgroup_size** (*int*) – Maximum workgroup size.
- **laser_scans_vertical_axis** (*int*) – Common laser scans axis.
- **match_laser_scans** (*bool*) – Match laser scans using geometric features.
- **progress** (*Callable[[float], None]*) – Progress callback.

mergeComponents(*components*[, *progress*])

Merge components.

Parameters

- **components** (*list[Metashape.Component]*) – List of components to merge.
- **progress** (*Callable[[float], None]*) – Progress callback.

meta

Chunk meta data.

Type
Metashape.MetaData

model

Default model for the current frame.

Type
Metashape.Model

model_group

Default model group for the current chunk.

Type
Metashape.ModelGroup

model_groups

List of model groups in the chunk.

Type
list[*Metashape.ModelGroup*]

models

List of models for the current frame.

Type
list[*Metashape.Model*]

modified

Modified flag.

Type
bool

optimizeCameras(*fit_f=True*, *fit_cx=True*, *fit_cy=True*, *fit_b1=False*, *fit_b2=False*, *fit_k1=True*, *fit_k2=True*, *fit_k3=True*, *fit_k4=False*, *fit_p1=True*, *fit_p2=True*, *fit_corrections=False*, *adaptive_fitting=False*, *tiepoint_covariance=False*[, *progress*])

Perform optimization of tie points / camera parameters.

Parameters

- **fit_f** (*bool*) – Enable optimization of focal length coefficient.
- **fit_cx** (*bool*) – Enable optimization of X principal point coordinates.
- **fit_cy** (*bool*) – Enable optimization of Y principal point coordinates.
- **fit_b1** (*bool*) – Enable optimization of aspect ratio.
- **fit_b2** (*bool*) – Enable optimization of skew coefficient.
- **fit_k1** (*bool*) – Enable optimization of k1 radial distortion coefficient.
- **fit_k2** (*bool*) – Enable optimization of k2 radial distortion coefficient.
- **fit_k3** (*bool*) – Enable optimization of k3 radial distortion coefficient.
- **fit_k4** (*bool*) – Enable optimization of k3 radial distortion coefficient.
- **fit_p1** (*bool*) – Enable optimization of p1 tangential distortion coefficient.
- **fit_p2** (*bool*) – Enable optimization of p2 tangential distortion coefficient.
- **fit_corrections** (*bool*) – Enable optimization of additional corrections.
- **adaptive_fitting** (*bool*) – Enable adaptive fitting of distortion coefficients.
- **tiepoint_covariance** (*bool*) – Estimate tie point covariance matrices.
- **progress** (*Callable[[float], None]*) – Progress callback.

orthomosaic

Default orthomosaic for the current frame.

Type
Metashape.Orthomosaic

orthomosaics

List of orthomosaics for the current frame.

Type
list[*Metashape.Orthomosaic*]

point_cloud

Default point cloud for the current frame.

Type
Metashape.PointCloud

point_cloud_groups

List of point cloud groups in the chunk.

Type
list[*Metashape.PointCloudGroup*]

point_clouds

List of point clouds for the current frame.

Type

list[*Metashape.PointCloud*]

primary_channel

Primary channel index (-1 for default).

Type

int

publishData(*service=ServiceSketchfab*, *source_data=TiePointsData*, *raster_transform=RasterTransformNone*, *save_point_color=True*, *save_camera_track=True*, *title=''*, *description=''*, *tags=''*, *owner=''*, *token=''*, *username=''*, *password=''*, *account=''*, *hostname=''*, *is_draft=False*, *is_private=False*, *is_protected=False*, *tile_size=256*, *min_zoom_level=-1*, *max_zoom_level=-1*[, *projection*], *resolution=0*[, *point_classes*][, *image_compression*][, *progress*])

Publish generated data online.

Parameters

- **service** (`Metashape.ServiceType`) – Service to upload on.
- **source_data** (`Metashape.DataSource`) – Asset type to upload.
- **raster_transform** (`Metashape.RasterTransformType`) – Raster band transformation.
- **save_point_color** (*bool*) – Enables/disables export of point colors.
- **save_camera_track** (*bool*) – Enables/disables export of camera track.
- **title** (*str*) – Dataset title.
- **description** (*str*) – Dataset description.
- **tags** (*str*) – Dataset tags.
- **owner** (*str*) – Account owner (Cesium and Mapbox services).
- **token** (*str*) – Account token (Cesium, Mapbox, Picterra, Pointbox and Sketchfab services).
- **username** (*str*) – Account username (4DMapper, Melown and Pointscene services).
- **password** (*str*) – Account password (4DMapper, Melown, Pointscene and Sketchfab services).
- **account** (*str*) – Account name (Melown service).
- **hostname** (*str*) – Service hostname (4DMapper service).
- **is_draft** (*bool*) – Mark dataset as draft (Sketchfab service).
- **is_private** (*bool*) – Set dataset access to private (Pointbox and Sketchfab services).
- **is_protected** (*bool*) – Set dataset access to protected (Pointbox service).
- **tile_size** (*int*) – Tile size in pixels.
- **min_zoom_level** (*int*) – Minimum zoom level.
- **max_zoom_level** (*int*) – Maximum zoom level.
- **projection** (`Metashape.CoordinateSystem`) – Output projection.

- **resolution** (*float*) – Output resolution in meters.
- **point_classes** (*list[int]*) – List of point classes to be exported.
- **image_compression** (Metashape.ImageCompression) – Image compression parameters.
- **progress** (*Callable[[float], None]*) – Progress callback.

raster_transform

Raster transform.

Type

Metashape.RasterTransform

reduceOverlap(*overlap=3*, *use_selection=False*[, *progress*])

Disable redundant cameras.

Parameters

- **overlap** (*int*) – Target number of cameras observing each point of the surface.
- **use_selection** (*bool*) – Focus on model selection.
- **progress** (*Callable[[float], None]*) – Progress callback.

refineMarkers([*markers*][, *progress*])

Refine markers based on images content.

Parameters

- **markers** (*list[int]*) – Optional list of markers to be processed.
- **progress** (*Callable[[float], None]*) – Progress callback.

refineModel(*downscale=4*, *iterations=10*, *smoothness=0.5*[, *cameras*][, *progress*])

Refine polygonal model.

Parameters

- **downscale** (*int*) – Refinement quality (1 - Ultra high, 2 - High, 4 - Medium, 8 - Low, 16 - Lowest).
- **iterations** (*int*) – Number of refinement iterations.
- **smoothness** (*float*) – Smoothing strength. Should be in range [0, 1].
- **cameras** (*list[int]*) – List of cameras to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

region

Reconstruction volume selection.

Type

Metashape.Region

remove(*items*)

Remove items from the chunk.

Parameters

items (*list[Metashape.Chunk | Metashape.Sensor | Metashape.CameraGroup | Metashape.MarkerGroup | Metashape.ScalebarGroup | Metashape.Camera | Metashape.Marker | Metashape.Scalebar | Metashape.CameraTrack]*) – A list of items to be removed.

removeLighting(*color_mode=False*, *internal_blur=1.5*, *mesh_noise_suppression=1*, *ambient_occlusion_path=''*, *ambient_occlusion_multiplier=1.5*[, *progress*])

Generate model for the chunk frame.

Parameters

- **color_mode** (*bool*) – Enable multi-color processing mode.
- **internal_blur** (*float*) – Internal blur. Should be in range [0, 4].
- **mesh_noise_suppression** (*float*) – Mesh normals noise suppression strength. Should be in range [0, 4].
- **ambient_occlusion_path** (*str*) – Path to ambient occlusion texture atlas. Can be empty.
- **ambient_occlusion_multiplier** (*float*) – Ambient occlusion multiplier. Should be in range [0.25, 4].
- **progress** (*Callable[[float], None]*) – Progress callback.

renderPreview(*width = 2048*, *height = 2048*[, *transform*], *point_size=1*[, *progress*])

Generate preview image for the chunk.

Parameters

- **width** (*int*) – Preview image width.
- **height** (*int*) – Preview image height.
- **transform** (*Metashape.Matrix*) – 4x4 viewpoint transformation matrix.
- **point_size** (*int*) – Point size.
- **progress** (*Callable[[float], None]*) – Progress callback.

Returns
Preview image.

Return type
Metashape.Image

resetRegion()

Reset reconstruction volume selector to default position.

scalebar_accuracy

Expected scale bar accuracy in meters.

Type
float

scalebar_groups

List of scale bar groups in the chunk.

Type
list[*Metashape.ScalebarGroup*]

scalebars

List of scale bars in the chunk.

Type
list[*Metashape.Scalebar*]

selected

Selects/deselects the chunk.

Type
bool

sensors

List of sensors in the chunk.

Type
list[*Metashape.Sensor*]

shapes

Shapes for the current frame.

Type
Metashape.Shapes

smoothModel(*strength=3*, *apply_to_selection=False*, *fix_borders=True*, *preserve_edges=False*[, *model*][, *progress*])

Smooth model using Laplacian smoothing algorithm.

Parameters

- **strength** (*float*) – Smoothing strength.
- **apply_to_selection** (*bool*) – Apply to selected faces.
- **fix_borders** (*bool*) – Fix borders.
- **preserve_edges** (*bool*) – Preserve edges.
- **model** (*int*) – Key of model to smooth.
- **progress** (*Callable[[float], None]*) – Progress callback.

smoothPointCloud(*smoothing_radius=0*[, *point_cloud*][, *classes*], *apply_to_selection=False*[, *progress*])

Smooth point cloud.

Parameters

- **smoothing_radius** (*float*) – Desired smoothing radius (m).
- **point_cloud** (*int*) – Key of point cloud to filter.
- **classes** (*list[int]*) – List of point classes to be smoothed.
- **apply_to_selection** (*bool*) – Smooth points within selection.
- **progress** (*Callable[[float], None]*) – Progress callback.

sortCameras()

Sorts cameras by their labels.

sortMarkers()

Sorts markers by their labels.

sortScalebars()

Sorts scalebars by their labels.

splitComponents(*items*[, *progress*])

Split components.

Parameters

- **items** (*list[*Metashape.Camera *|* Metashape.PointCloud*]*) – List of items to split.
- **progress** (*Callable[[float], None]*) – Progress callback.

thinTiePoints(*point_limit=1000*)

Remove excessive tracks from the tie point cloud.

Parameters

point_limit (*int*) – Maximum number of points for each photo.

thumbnails

Image thumbnails.

Type

Metashape.Thumbnails

tie_points

Generated tie point cloud.

Type

Metashape.TiePoints

tiepoint_accuracy

Expected tie point accuracy in pixels.

Type

float

tiled_model

Default tiled model for the current frame.

Type

Metashape.TiledModel

tiled_models

List of tiled models for the current frame.

Type

list[*Metashape.TiledModel*]

trackMarkers(*first_frame=0*, *last_frame=0*[, *progress*])

Track marker projections through the frame sequence.

Parameters

- **first_frame** (*int*) – Starting frame index.
- **last_frame** (*int*) – Ending frame index.
- **progress** (*Callable[[float], None]*) – Progress callback.

transform

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

Type

Metashape.ChunkTransform

transformRaster(*source_data=ElevationData*[, *asset*], *subtract=False*[, *operand_chunk*][, *operand_frame*][, *operand_asset*], *width=0*, *height=0*[, *world_transform*], *resolution=0*, *resolution_x=0*, *resolution_y=0*, *nodata_value=-32767*, *north_up=True*[, *region*][, *projection*], *replace_asset=False*[, *frames*][, *progress*])

Transform DEM or orthomosaic.

Parameters

- **source_data** (`Metashape.DataSource`) – Selects between DEM and orthomosaic.
- **asset** (*int*) – Asset key to transform.
- **subtract** (*bool*) – Subtraction flag.
- **operand_chunk** (*int*) – Operand chunk key.
- **operand_frame** (*int*) – Operand frame key.
- **operand_asset** (*int*) – Operand asset key.
- **width** (*int*) – Raster width.
- **height** (*int*) – Raster height.
- **world_transform** (`Metashape.Matrix`) – 2x3 raster-to-world transformation matrix.
- **resolution** (*float*) – Output resolution in meters.
- **resolution_x** (*float*) – Pixel size in the X dimension in projected units.
- **resolution_y** (*float*) – Pixel size in the Y dimension in projected units.
- **nodata_value** (*float*) – No-data value (DEM export only).
- **north_up** (*bool*) – Use north-up orientation for export.
- **region** (`Metashape.BBox`) – Region to be processed.
- **projection** (`Metashape.OrthoProjection`) – Output projection.
- **replace_asset** (*bool*) – Replace default raster with transformed one.
- **frames** (*list[int]*) – List of frames to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

triangulateTiePoints(*max_error=10*, *min_image=2*[, *progress*])

Rebuild tie point cloud for the chunk.

Parameters

- **max_error** (*float*) – Reprojection error threshold.
- **min_image** (*int*) – Minimum number of point projections.
- **progress** (*Callable[[float], None]*) – Progress callback.

updateTransform()

Update chunk transformation based on reference data.

world_crs

Coordinate system used as world coordinate system.

Type

Metashape.CoordinateSystem

class Metashape.**ChunkTransform**

Transformation between chunk and world coordinates systems.

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.ChunkTransform

matrix

Transformation matrix.

Type
Metashape.Matrix

rotation

Rotation component.

Type
Metashape.Matrix

scale

Scale component.

Type
float

translation

Translation component.

Type
Metashape.Vector

class Metashape.**CirTransform**

CIR calibration matrix.

calibrate()

Calibrate CIR matrix based on orthomosaic histogram.

coeffs

Color matrix.

Type
Metashape.Matrix

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.CirTransform

reset()

Reset CIR calibration matrix.

class Metashape.**ClassificationMethod**

Index values classification method in [EqualIntervalsClassification, JenksNaturalBreaksClassification]

class Metashape.**CloudClient**

CloudClient class provides access to the Agisoft Cloud processing service and allows to create and manage cloud projects.

The following example connects to the service and lists available projects:

```
>>> import Metashape
>>> client = Metashape.CloudClient()
>>> client.username = 'user'
>>> client.password = 'password'
>>> client.projectList()
```

abortProcessing(*document*)

Cancel processing.

Parameters
document (Metashape.Document) – Project to cancel.

client_id

Client software id (optional).

Type
str

client_secret

Client softrwae secret (optional).

Type
str

downloadProject(*document*[, *progress*])

Download project from the cloud.

Parameters

- **document** (Metashape.Document) – Project to download.
- **progress** (*Callable[[float], None]*) – Progress callback.

getProcessingStatus(*document*)

Get processing status.

Parameters
document (Metashape.Document) – Project being processed.

Returns
Processing status.

Return type
dict

getProjectList()

Get list of projects in the cloud.

Returns
List of projects.

Return type
list

password
Cloud account password.

Type
str

processProject(*document*, *tasks*)
Start processing in the cloud.

Parameters

- **document** (`Metashape.Document`) – Project to process.
- **tasks** (*list[*`Metashape.NetworkTask`*]*) – List of processing tasks to execute.

uploadProject(*document*, *publish=False*[, *progress*])
Upload project to the cloud.

Parameters

- **document** (`Metashape.Document`) – Project to upload.
- **publish** (*bool*) – Publish project for online visualization.
- **progress** (*Callable[[float], None]*) – Progress callback.

username
Cloud account username.

Type
str

class Metashape.Component
Component instance

chunk
Chunk the component belongs to.

Type
Metashape.Chunk

key
Component identifier.

Type
int

label
Component label.

Type
str

partition
Component partition.

Type
list

region

Reconstruction volume selection.

Type
Metashape.Region

transform

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

Type
Metashape.ChunkTransform

class Metashape.**CoordinateSystem**

Coordinate reference system (local, geographic or projected).

The following example changes chunk coordinate system to WGS 84 / UTM zone 41N and loads reference data from file:

```
>>> import Metashape
>>> chunk = Metashape.app.document.chunk
>>> chunk.crs = Metashape.CoordinateSystem("EPSG::32641")
>>> chunk.importReference("gcp.txt", Metashape.ReferenceFormatCSV)
>>> chunk.updateTransform()
```

addGeoid(*path*)

Register geoid model.

Parameters
path (*str*) – Path to geoid file.

authority

Authority identifier of the coordinate system.

Type
str

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.CoordinateSystem

datumTransform(*source*, *target*)

Coordinate transformation from source to target coordinate system datum.

Parameters

- **source** (Metashape.CoordinateSystem) – Source coordinate system.
- **target** (Metashape.CoordinateSystem) – Target coordinate system.

Returns
4x4 transformation matrix.

Return type
Metashape.Matrix

geoccs

Base geocentric coordinate system.

Type
Metashape.CoordinateSystem

geogcs

Base geographic coordinate system.

Type
Metashape.CoordinateSystem

geoid_height

Fixed geoid height to be used instead of interpolated values.

Type
float

init(*crs*)

Initialize projection based on specified WKT definition or authority identifier.

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

listBuiltinCRS()

Returns a list of builtin coordinate systems.

localframe(*point*)

Returns 4x4 transformation matrix to LSE coordinates at the given point.

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

Returns
Transformation from geocentric coordinates to local coordinates.

Return type
Metashape.Matrix

name

Name of the coordinate system.

Type
str

proj4

Coordinate system definition in PROJ.4 format.

Type
str

project(*point*)

Projects point from geocentric coordinates to projected geographic coordinate system.

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

Returns
3D point in projected coordinates.

Return type
Metashape.Vector

towgs84

TOWGS84 transformation parameters (dx, dy, dz, rx, ry, rz, scale).

Type
list[float]

transform(*point*, *source*, *target*)

Transform point coordinates between coordinate systems.

Parameters

- **point** (Metashape.Vector) – 2D or 3D point coordinates.
- **source** (Metashape.CoordinateSystem) – Source coordinate system.
- **target** (Metashape.CoordinateSystem) – Target coordinate system.

Returns
Transformed point coordinates.

Return type
Metashape.Vector

transformationMatrix(*point*, *source*, *target*)

Local approximation of coordinate transformation from source to target coordinate system at the given point.

Parameters

- **point** (Metashape.Vector) – 3D point coordinates.
- **source** (Metashape.CoordinateSystem) – Source coordinate system.
- **target** (Metashape.CoordinateSystem) – Target coordinate system.

Returns
4x4 transformation matrix.

Return type
Metashape.Matrix

unproject(*point*)

Unprojects point from projected coordinates to geocentric coordinates.

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

Returns
3D point in geocentric coordinates.

Return type
Metashape.Vector

wkt

Coordinate system definition in WKT format.

Type
str

wkt2

Coordinate system definition in WKT format, version 2.

Type
str

class Metashape.**DataSource**

Data source in [TiePointsData, PointCloudData, ModelData, TiledModelData, ElevationData, OrthomosaicData, DepthMapsData, ImagesData, TrajectoryData, LaserScansData, DepthMapsAndLaserScansData]

class Metashape.**DataType**

Data type in [DataTypeUndefined, DataType8i, DataType8u, DataType16i, DataType16u, DataType16f, DataType32i, DataType32u, DataType32f, DataType64i, DataType64u, DataType64f]

class Metashape.**DepthMap**

Depth map data.

calibration

Depth map calibration.

Type
Metashape.Calibration

copy()

Returns a copy of the depth map.

Returns
Copy of the depth map.

Return type
Metashape.DepthMap

getCalibration(*level=0*)

Returns calibration data.

Parameters
level (*int*) – Level index.

Returns
Calibration data.

Return type
Metashape.Calibration

image([*level*])

Returns image data.

Parameters
level (*int*) – Level index.

Returns
Image data.

Return type
Metashape.Image

setCalibration(*calibration*, *level=0*)

Parameters

- **calibration** (`Metashape.Calibration`) – Calibration data.
- **level** (*int*) – Level index.

setImage(*image*, *level=0*)

Parameters

- **image** (`Metashape.Image`) – Image object with depth map data.

- **level** (*int*) – Level index.

class Metashape.**DepthMaps**

A set of depth maps generated for a chunk frame.

clear()

Clears depth maps data.

copy()

Create a copy of the depth maps.

Returns
Copy of the depth maps.

Return type
Metashape.DepthMaps

items()

List of items.

key

Depth maps identifier.

Type
int

keys()

List of item keys.

label

Depth maps label.

Type
str

meta

Depth maps meta data.

Type
Metashape.MetaData

modified

Modified flag.

Type
bool

values()

List of item values.

class Metashape.**Document**

Metashape project.

Contains list of chunks available in the project. Implements processing operations that work with multiple chunks. Supports saving/loading project files.

The project currently opened in Metashape window can be accessed using Metashape.app.document attribute. Additional Document objects can be created as needed.

The following example saves active chunk from the opened project in a separate project:

```
>>> import Metashape
>>> doc = Metashape.app.document
>>> doc.save(path = "project.psz", chunks = [doc.chunk])
```

addChunk()

Add new chunk to the document.

Returns
Created chunk.

Return type
Metashape.Chunk

alignChunks([*chunks*][, *reference*], *method=0*, *fit_scale=True*, *downscale=1*, *generic_preselection=False*, *filter_mask=False*, *mask_tiepoints=False*, *keypoint_limit=40000*[, *markers*][, *progress*])

Align specified set of chunks.

Parameters

- **chunks** (*list[int]*) – List of chunks to be aligned.
- **reference** (*int*) – Chunk to be used as a reference.
- **method** (*int*) – Alignment method (0 - point based, 1 - marker based, 2 - camera based).
- **fit_scale** (*bool*) – Fit chunk scale during alignment.
- **downscale** (*int*) – Alignment accuracy (0 - Highest, 1 - High, 2 - Medium, 4 - Low, 8 - Lowest).
- **generic_preselection** (*bool*) – Enables image pair preselection.
- **filter_mask** (*bool*) – Filter points by mask.
- **mask_tiepoints** (*bool*) – Apply mask filter to tie points.
- **keypoint_limit** (*int*) – Maximum number of points for each photo.
- **markers** (*list[int]*) – List of markers to be used for marker based alignment.
- **progress** (*Callable[[float], None]*) – Progress callback.

append(*document*[, *chunks*][, *progress*])

Append the specified Document object to the current document.

Parameters

- **document** (Metashape.Document) – Document object to be appended.
- **chunks** (*list[Metashape.Chunk]*) – List of chunks to append.
- **progress** (*Callable[[float], None]*) – Progress callback.

chunk

Active chunk.

Type
Metashape.Chunk

chunks

List of chunks in the document.

Type
list[*Metashape.Chunk*]

clear()

Clear the contents of the Document object.

copy()

Return a copy of the document.

Returns
A copy of the document.

Return type
Metashape.Document

findChunk(*key*)

Find chunk by its key.

Returns
Found chunk.

Return type
Metashape.Chunk

mergeChunks(*copy_laser_scans=True*, *copy_depth_maps=False*, *copy_point_clouds=False*, *copy_models=False*, *copy_tiled_models=False*, *copy_elevations=False*, *copy_orthomosaics=False*, *merge_markers=False*, *merge_tiepoints=False*, *merge_assets=False*[, *chunks*][, *progress*])

Merge specified set of chunks.

Parameters

- **copy_laser_scans** (*bool*) – Copy laser scans.
- **copy_depth_maps** (*bool*) – Copy depth maps.
- **copy_point_clouds** (*bool*) – Copy point clouds.
- **copy_models** (*bool*) – Copy models.
- **copy_tiled_models** (*bool*) – Copy tiled models.
- **copy_elevations** (*bool*) – Copy DEMs.
- **copy_orthomosaics** (*bool*) – Copy orthomosaics.
- **merge_markers** (*bool*) – Merge markers.
- **merge_tiepoints** (*bool*) – Merge tie points.
- **merge_assets** (*bool*) – Merge default assets.
- **chunks** (*list[int]*) – List of chunks to process.
- **progress** (*Callable[[float], None]*) – Progress callback.

meta

Document meta data.

Type
Metashape.MetaData

modified

Modified flag.

Type
bool

open(*path*, *read_only=False*, *ignore_lock=False*, *archive=True*)

Load document from the specified file.

Parameters

- **path** (*str*) – Path to the file.
- **read_only** (*bool*) – Open document in read-only mode.
- **ignore_lock** (*bool*) – Ignore lock state for project modifications.
- **archive** (*bool*) – Override project format when using non-standard file extension.

path

Path to the document file.

Type
str

read_only

Read only status.

Type
bool

remove(*items*)

Remove a set of items from the document.

Parameters
items (*list[Metashape.Chunk]*) – A list of items to be removed.

save([*path*][, *chunks*][, *version*], *archive=True*)

Save document to the specified file.

Parameters

- **path** (*str*) – Optional path to the file.
- **chunks** (*list[Metashape.Chunk]*) – List of chunks to be saved.
- **version** (*str*) – Project version to save.
- **archive** (*bool*) – Override project format when using non-standard file extension.

class Metashape.Elevation

Digital elevation model.

altitude(*point*)

Return elevation value at the specified point.

Parameters
point (*Metashape.Vector*) – Point coordinates in the elevation coordinate system.

Returns
Elevation value.

Return type
float

bottom

Y coordinate of the bottom side.

Type
float

clear()
Clears elevation model data.

copy()
Create a copy of the elevation model.

Returns
Copy of the elevation model.

Return type
Metashape.Elevation

crs
Coordinate system of elevation model.

Type
Metashape.CoordinateSystem

height
Elevation model height.

Type
int

key
Elevation model identifier.

Type
int

label
Elevation model label.

Type
str

left
X coordinate of the left side.

Type
float

max
Maximum elevation value.

Type
float

meta
Elevation model meta data.

Type
Metashape.MetaData

min
Minimum elevation value.

Type
float

modified

Modified flag.

Type
bool

palette

Color palette.

Type
dict

pickPoint(*origin*, *target*)

Returns ray intersection with the DEM (point on the ray nearest to some point).

Parameters

- **origin** (`Metashape.Vector`) – Ray origin in the DEM coordinate system.
- **target** (`Metashape.Vector`) – Point on the ray in the DEM coordinate system.

Returns
Coordinates of the intersection point in the DEM coordinate system.

Return type
Metashape.Vector

projection

Projection of elevation model.

Type
Metashape.OrthoProjection

resolution

DEM resolution in meters.

Type
float

right

X coordinate of the right side.

Type
float

top

Y coordinate of the top side.

Type
float

width

Elevation model width.

Type
int

class Metashape.**EulerAngles**

Euler angles in [EulerAnglesYPR, EulerAnglesOPK, EulerAnglesPOK, EulerAnglesANK]

class Metashape.**FaceCount**

Face count in [LowFaceCount, MediumFaceCount, HighFaceCount, CustomFaceCount]

class Metashape.**FilterMode**

Depth filtering mode in [NoFiltering, MildFiltering, ModerateFiltering, AggressiveFiltering]

class Metashape.**FrameStep**

Frame step size for video import in [CustomFrameStep, SmallFrameStep, MediumFrameStep, LargeFrameStep]

class Metashape.**Geometry**

Geometry data.

GeometryCollection(*geometries*)

Create a GeometryCollection geometry.

Parameters
geometries (*list[Metashape.Geometry]*) – Child geometries.

Returns
A GeometryCollection geometry.

Return type
Metashape.Geometry

LineString(*coordinates*)

Create a LineString geometry.

Parameters
coordinates (*list[Metashape.Vector]*) – List of vertex coordinates.

Returns
A LineString geometry.

Return type
Metashape.Geometry

MultiLineString(*geometries*)

Create a MultiLineString geometry.

Parameters
geometries (*list[Metashape.Geometry]*) – Child line strings.

Returns
A point geometry.

Return type
Metashape.Geometry

MultiPoint(*geometries*)

Create a MultiPoint geometry.

Parameters
geometries (*list[Metashape.Geometry]*) – Child points.

Returns
A point geometry.

Return type
Metashape.Geometry

MultiPolygon(*geometries*)

Create a MultiPolygon geometry.

Parameters
geometries (*list[Metashape.Geometry]*) – Child polygons.

Returns
A point geometry.

Return type
Metashape.Geometry

`Point(vector)`

Create a Point geometry.

Parameters
vector (`Metashape.Vector | list[float]`) – Point coordinates.

Returns
A point geometry.

Return type
Metashape.Geometry

`Polygon(exterior_ring[, interior_rings ])`

Create a Polygon geometry.

Parameters

- **exterior_ring** (`list[Metashape.Vector]`) – Point coordinates.
- **interior_rings** (`list[Metashape.Vector]`) – Point coordinates.

Returns
A Polygon geometry.

Return type
Metashape.Geometry

`class Type`

Geometry type in [PointType, LineStringType, PolygonType, MultiPointType, MultiLineStringType, MultiPolygonType, GeometryCollectionType]

`coordinates`

List of vertex coordinates.

Type
list[*Metashape.Vector*]

`geometries`

List of child geometries.

Type
list[*Metashape.Geometry*]

`is_3d`

Is 3D flag.

Type
bool

`type`

Geometry type.

Type
Metashape.Geometry.Type

class Metashape.**Image**(*width*, *height*, *channels*, *datatype='U8'*)

n-channel image

Parameters

- **width** (*int*) – image width
- **height** (*int*) – image height
- **channels** (*str*) – color channel layout, e.g. 'RGB', 'RGBA', etc.
- **datatype** (*str*) – pixel data type in ['U8', 'U16', 'U32', 'F16', 'F32', 'F64']

channels

Channel mapping for the image.

Type
str

cn

Number of color channels.

Type
int

convert(*channels*[, *datatype*])

Convert image to specified data type and channel layout.

Parameters

- **channels** (*str*) – color channels to be loaded, e.g. 'RGB', 'RGBA', etc.
- **datatype** (*str*) – pixel data type in ['U8', 'U16', 'U32', 'F16', 'F32', 'F64']

Returns
Converted image.

Return type
Metashape.Image

copy()

Return a copy of the image.

Returns
copy of the image

Return type
Metashape.Image

data_type

Data type used to store pixel values.

Type
str

fromstring(*data*, *width*, *height*, *channels*, *datatype='U8'*)

Create image from byte array.

Parameters

- **data** (*str*) – raw image data
- **width** (*int*) – image width
- **height** (*int*) – image height

- **channels** (*str*) – color channel layout, e.g. 'RGB', 'RGBA', etc.
- **datatype** (*str*) – pixel data type in ['U8', 'U16', 'U32', 'F16', 'F32', 'F64']

Returns
Created image.

Return type
Metashape.Image

`gaussianBlur`(*radius*)

Smooth image with a gaussian filter.

Parameters
radius (*float*) – smoothing radius.

Returns
Smoothed image.

Return type
Metashape.Image

`height`

Image height.

Type
int

`open`(*path*, *layer=0*, *datatype='U8'*[, *channels*][, *x*][, *y*][, *w*][, *h*])

Load image from file.

Parameters

- **path** (*str*) – path to the image file
- **layer** (*int*) – image layer in case of multipage file
- **datatype** (*str*) – pixel data type in ['U8', 'U16', 'U32', 'F16', 'F32', 'F64']
- **channels** (*str*) – color channels to be loaded, e.g. 'RGB', 'RGBA', etc.
- **x** (*int*) – x offset of image region.
- **y** (*int*) – y offset of image region.
- **w** (*int*) – width of image region.
- **h** (*int*) – height of image region.

Returns
Loaded image.

Return type
Metashape.Image

`resize`(*width*, *height*)

Resize image to specified dimensions.

Parameters

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

Returns
resized image

Return type
Metashape.Image

save(*path*[, *compression*])
Save image to the file.

Parameters
- **path** (*str*) – path to the image file
- **compression** (Metashape.ImageCompression) – compression options

tostring()
Convert image to byte array.

Returns
Raw image data.

Return type
str

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

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

Returns
undistorted image

Return type
Metashape.Image

uniformNoise(*amplitude*)
Add uniform noise with specified amplitude.

Parameters
amplitude (*float*) – noise amplitude.

Returns
Image with added noise.

Return type
Metashape.Image

warp(*calib0*, *trans0*, *calib1*, *trans1*)
Warp image by rotating virtual viewpoint.

Parameters
- **calib0** (Metashape.Calibration) – initial calibration
- **trans0** (Metashape.Matrix) – initial camera orientation as 4x4 matrix
- **calib1** (Metashape.Calibration) – final calibration
- **trans1** (Metashape.Matrix) – final camera orientation as 4x4 matrix

Returns
warped image

Return type
Metashape.Image

width

Image width.

Type
int

class Metashape.ImageCompression

Image compression parameters

class TiffCompression

Tiff compression in [TiffCompressionNone, TiffCompressionLZW, TiffCompressionJPEG, TiffCompressionPackbits, TiffCompressionDeflate]

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.Viewpoint

jpeg_quality

JPEG quality.

Type
int

tiff_big

Enable BigTIFF compression for TIFF files.

Type
bool

tiff_compression

Tiff compression.

Type
int

tiff_overviews

Enable image pyramid deneration for TIFF files.

Type
bool

tiff_tiled

Export tiled TIFF.

Type
bool

class Metashape.ImageFormat

Image format in [ImageFormatNone, ImageFormatJPEG, ImageFormatTIFF, ImageFormatPNG, ImageFormatBMP, ImageFormatEXR, ImageFormatPNM, ImageFormatSGI, ImageFormatCR2, ImageFormatBZ2, ImageFormatSEQ, ImageFormatBIL, ImageFormatASCII, ImageFormatXYZ, ImageFormatARA, ImageFormatTGA, ImageFormatDDS, ImageFormatJP2, ImageFormatWebP, ImageFormatJXL, ImageFormatKTX]

class Metashape.**ImageLayout**

Image layout in [UndefinedLayout, FlatLayout, MultiframeLayout, MultiplaneLayout]

class Metashape.**Interpolation**

Interpolation mode in [DisabledInterpolation, EnabledInterpolation, Extrapolated]

class Metashape.**License**

License information.

activate(*license_key*)

Activate software online using a license key.

Parameters
key (*str*) – Activation key.

activateOffline(*activation_params*)

Create a request for offline activation.

Parameters
activation_params (*str*) – The content of .actparam file.

Returns
The activation request which should be saved to .actreq file.

Return type
str

borrowLicense(*seconds*)

Borrow floating license for the specified number of seconds.

Parameters
seconds (*int*) – Borrow duration in seconds.

deactivate()

Deactivate software online.

deactivateOffline()

Create a request for offline deactivation.

Returns
The deactivation request which should be saved to .actreq file.

Return type
str

expiration

License expiration as a Unix timestamp in seconds.

Type
int

install(*activation_response*)

Install license from the activation response.

Parameters
activation_response (*str*) – The content of .actresp file.

returnLicense()

Return borrowed license to the license server.

valid

Metashape activation status.

Type
bool

class Metashape.MappingMode

UV mapping mode in [GenericMapping, OrthophotoMapping, AdaptiveOrthophotoMapping, SphericalMapping, CameraMapping]

class Metashape.Marker

Marker instance

class Projection

Marker data().

coord

Point coordinates in pixels.

Type
Metashape.Vector

pinned

Pinned flag.

Type
bool

valid

Valid flag.

Type
bool

class Projections

Collection of projections specified for the marker

items()

List of items.

keys()

List of item keys.

values()

List of item values.

class Reference

Marker reference data.

accuracy

Marker location accuracy.

Type
Metashape.Vector

enabled

Enabled flag.

Type
bool

location
Marker coordinates.

Type
Metashape.Vector

class Type
Marker type in [Regular, Vertex, Fiducial]

chunk
Chunk the marker belongs to.

Type
Metashape.Chunk

enabled
Enables/disables the marker.

Type
bool

frames
Marker frames.

Type
list[*Metashape.Marker*]

group
Marker group.

Type
Metashape.MarkerGroup

key
Marker identifier.

Type
int

label
Marker label.

Type
str

meta
Marker meta data.

Type
Metashape.MetaData

position
Marker position in the current frame.

Type
Metashape.Vector

position_covariance
Marker position covariance.

Type
Metashape.Matrix

projections

List of marker projections.

Type
Metashape.Marker.Projections

reference

Marker reference data.

Type
Metashape.Marker.Reference

selected

Selects/deselects the marker.

Type
bool

sensor

Fiducial mark sensor.

Type
Metashape.Sensor

type

Marker type.

Type
Metashape.Marker.Type

class Metashape.MarkerGroup

MarkerGroup objects define groups of multiple markers. The grouping is established by assignment of a MarkerGroup instance to the Marker.group attribute of participating markers.

label

Marker group label.

Type
str

selected

Current selection state.

Type
bool

class Metashape.Mask

Mask instance

copy()

Returns a copy of the mask.

Returns
Copy of the mask.

Return type
Metashape.Mask

image()

Returns image data.

Returns
Image data.

Return type
Metashape.Image

invert()

Create inverted copy of the mask.

Returns
Inverted copy of the mask.

Return type
Metashape.Mask

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

Loads mask from file.

Parameters

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

setImage(*image*)

Parameters
image (Metashape.Image) – Image object with mask data.

class Metashape.**MaskOperation**

Mask operation in [MaskOperationReplacement, MaskOperationUnion, MaskOperationIntersection, MaskOperationDifference]

class Metashape.**MaskingMode**

Masking mode in [MaskingModeAlpha, MaskingModeFile, MaskingModeBackground, MaskingModeModel]

class Metashape.**Masks**

A set of masks for a chunk frame.

items()

List of items.

keys()

List of item keys.

meta

Thumbnails meta data.

Type
Metashape.MetaData

modified

Modified flag.

Type
bool

values()

List of item values.

class Metashape.**Matrix**

m-by-n matrix

```
>>> import Metashape
>>> m1 = Metashape.Matrix.Diag( (1,2,3,4) )
>>> m3 = Metashape.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:
...     Metashape.app.messageBox("Diagonal matrix dimensions: " + str(m3.size))
```

Diag(*vector*)

Create a diagonal matrix.

Parameters
`vector` (`Metashape.Vector | list[float]`) – The vector of diagonal entries.

Returns
A diagonal matrix.

Return type
Metashape.Matrix

Rotation(*matrix*)

Create a rotation matrix.

Parameters
`matrix` (`Metashape.Matrix`) – The 3x3 rotation matrix.

Returns
4x4 matrix representing rotation.

Return type
Metashape.Matrix

Scale(*scale*)

Create a scale matrix.

Parameters
`scale` (`Metashape.Vector`) – The scale vector.

Returns
A matrix representing scale.

Return type
Metashape.Matrix

Translation(*vector*)

Create a translation matrix.

Parameters
`vector` (`Metashape.Vector`) – The translation vector.

Returns
A matrix representing translation.

Return type
Metashape.Matrix

col(*index*)

Returns column of the matrix.

Returns
matrix column.

Return type
Metashape.Vector

copy()

Returns a copy of this matrix.

Returns
an instance of itself

Return type
Metashape.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
Metashape.Matrix

mulp(*point*)

Transforms a point in homogeneous coordinates.

Parameters
point (`Metashape.Vector`) – The point to be transformed.

Returns
transformed point.

Return type
Metashape.Vector

mulv(*vector*)

Transforms vector in homogeneous coordinates.

Parameters
vector (`Metashape.Vector`) – The vector to be transformed.

Returns
transformed vector.

Return type
Metashape.Vector

`rotation()`

Returns rotation component of the 4x4 matrix.

Returns
rotation component

Return type
Metashape.Matrix

`row(index)`

Returns row of the matrix.

Returns
matrix row.

Return type
Metashape.Vector

`scale()`

Returns scale component of the 4x4 matrix.

Returns
scale component

Return type
float

`size`

Matrix dimensions.

Type
tuple

`svd()`

Returns singular value decomposition of the matrix.

Returns
u, s, v tuple where a = u * diag(s) * v

Return type
tuple[*Metashape.Matrix*, *Metashape.Vector*, *Metashape.Matrix*]

`t()`

Return a new, transposed matrix.

Returns
a transposed matrix

Return type
Metashape.Matrix

`translation()`

Returns translation component of the 4x4 matrix.

Returns
translation component

Return type
Metashape.Vector

`zero()`

Set all matrix elements to zero.

class Metashape.**MetaData**(*object*)

Collection of object properties

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.MetaData

items()

List of items.

keys()

List of item keys.

values()

List of item values.

class Metashape.**Model**

Triangular mesh model instance

class Face

Triangular face of the model

hidden

Face visibility flag.

Type
bool

selected

Face selection flag.

Type
bool

tex_index

Texture page index.

Type
int

tex_vertices

Texture vertex indices.

Type
tuple[int, int, int]

vertices

Vertex indices.

Type
tuple[int, int, int]

class Faces

Collection of model faces

resize(*count*)

Resize faces list.

Parameters
`count` (*int*) – new face count

`class Statistics`

Model statistics

`components`

Number of connected components.

Type
int

`degenerate_faces`

Number of degenerate faces.

Type
int

`duplicate_faces`

Number of duplicate faces.

Type
int

`faces`

Total number of faces.

Type
int

`flipped_normals`

Number of edges with flipped normals.

Type
int

`free_vertices`

Number of free vertices.

Type
int

`invalid_vertices`

Number of vertices with NaN coordinates.

Type
int

`multiple_edges`

Number of edges connecting more than 2 faces.

Type
int

`open_edges`

Number of open edges.

Type
int

`out_of_range_indices`

Number of out of range indices.

Type
int

similar_vertices

Number of similar vertices.

Type
int

vertices

Total number of vertices.

Type
int

zero_faces

Number of zero faces.

Type
int

class TexVertex

Texture vertex of the model

coord

2D vertex coordinates.

Type
Metashape.Vector

class TexVertices

Collection of model texture vertices

resize(*count*)

Resize vertex list.

Parameters
count (*int*) – new vertex count

class Texture

Model texture.

image(*page=0*)

Return texture image.

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

Returns
Texture image.

Return type
Metashape.Image

label

Animation label.

Type
str

meta

Camera track meta data.

Type
Metashape.MetaData

model

Model the texture belongs to.

Type
Metashape.Model

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

Initialize texture from image data.

Parameters

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

type

Texture type.

Type

Metashape.Model.TextureType

class TextureType

Texture type in [DiffuseMap, NormalMap, OcclusionMap, DisplacementMap]

class Vertex

Vertex of the model

color

Vertex color.

Type

tuple of numbers

confidence

Vertex confidence.

Type

float

coord

Vertex coordinates.

Type

Metashape.Vector

class Vertices

Collection of model vertices

resize(*count*)

Resize vertex list.

Parameters

count (*int*) – new vertex count

addTexture(*type=Model.DiffuseMap*)

Add new texture to the model.

Parameters

type (Metashape.Model.TextureType) – Texture type.

Returns

Created texture.

Return type

Metashape.Model.Texture

area()

Return area of the model surface.

Returns

Model area.

Return type
float

`bands`
List of color bands.

Type
list[str]

`clear()`
Clears model data.

`closeHoles(level=30, apply_to_selection=False)`
Fill holes in the model surface.

Parameters

- `level` (*int*) – Hole size threshold in percents.
- `apply_to_selection` (*bool*) – Close holes within selection

`copy()`
Create a copy of the model.

Returns
Copy of the model.

Return type
Metashape.Model

`cropSelection()`
Crop selected faces and free vertices from the mesh.

`crs`
Reference coordinate system.

Type
Metashape.CoordinateSystem | None

`data_type`
Data type used to store color values.

Type
Metashape.DataType

`faces`
Collection of model faces.

Type
Metashape.Model.Faces

`fixTopology()`
Remove polygons causing topological problems.

`getActiveTexture(type=Model.DiffuseMap)`
Return active texture.

Parameters
`type` (`Metashape.Model.TextureType`) – Texture type.

Returns
Texture image.

Return type
Metashape.Image

group
Model group.

Type
Metashape.ModelGroup

key
Model identifier.

Type
int

label
Model label.

Type
str

loadTexture(*path*)
Load texture from the specified file.

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

meta
Model meta data.

Type
Metashape.MetaData

modified
Modified flag.

Type
bool

pickPoint(*origin*, *target*, *endpoints=1*)
Return ray intersection with mesh.

Parameters

- **origin** (Metashape.Vector) – Ray origin.
- **target** (Metashape.Vector) – Point on the ray.
- **endpoints** (*int*) – Number of endpoints to check for (0 - line, 1 - ray, 2 - segment).

Returns
Coordinates of the intersection point.

Return type
Metashape.Vector

remove(*items*)
Remove textures from the model.

Parameters
items (*list[Metashape.Model.Texture]*) – A list of textures to be removed.

removeComponents(*size*)

Remove small connected components.

Parameters
size (*int*) – Threshold on the polygon count of the components to be removed.

removeSelection()

Remove selected faces and free vertices from the mesh.

removeTextures()

Remove textures.

removeUV()

Remove UV mapping.

removeVertexColors()

Remove vertex colors.

removeVertexConfidence()

Remove confidence.

renderDepth(*transform*, *calibration*, *cull_faces=True*, *add_alpha=True*)

Render model depth image for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **cull_faces** (*bool*) – Enable back-face culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.

Returns
Rendered image.

Return type
Metashape.Image

renderImage(*transform*, *calibration*, *cull_faces=True*, *add_alpha=True*, *raster_transform=RasterTransformNone*)

Render model image for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **cull_faces** (*bool*) – Enable back-face culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.
- **raster_transform** (Metashape.RasterTransformType) – Raster band transformation.

Returns
Rendered image.

Return type
Metashape.Image

renderMask(*transform*, *calibration*, *cull_faces=True*)

Render model mask image for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **cull_faces** (*bool*) – Enable back-face culling.

Returns

Rendered image.

Return type

Metashape.Image

renderNormalMap(*transform*, *calibration*, *cull_faces=True*, *add_alpha=True*)

Render image with model normals for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **cull_faces** (*bool*) – Enable back-face culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.

Returns

Rendered image.

Return type

Metashape.Image

renderPreview(*width = 2048*, *height = 2048*[, *transform*][, *progress*])

Generate model preview image.

Parameters

- **width** (*int*) – Preview image width.
- **height** (*int*) – Preview image height.
- **transform** (Metashape.Matrix) – 4x4 viewpoint transformation matrix.
- **progress** (*Callable[[float], None]*) – Progress callback.

Returns

Preview image.

Return type

Metashape.Image

saveTexture(*path*)

Save texture to the specified file.

Parameters

path (*str*) – Path to the image file.

setActiveTexture(*texture*, *type=Model.DiffuseMap*)

Set active texture.

Parameters

- **texture** (Metashape.Model.Texture) – Texture to set.
- **type** (Metashape.Model.TextureType) – Texture type.

statistics([*progress*])

Return model statistics.

Parameters
progress (*Callable[[float], None]*) – Progress callback.

Returns
Model statistics.

Return type
Metashape.Model.Statistics

tex_vertices

Collection of model texture vertices.

Type
Metashape.Model.TexVertices

textures

List of model textures.

Type
list[*Metashape.Model.Texture*]

transform

4x4 model transformation matrix.

Type
Metashape.Matrix

transformVertices(*transform*)

Transform vertex coordinates.

Parameters
transform (Metashape.Matrix) – 4x4 transformation matrix.

vertices

Collection of model vertices.

Type
Metashape.Model.Vertices

volume()

Return volume of the closed model surface.

Returns
Model volume.

Return type
float

class Metashape.**ModelFormat**

Model format in [ModelFormatNone, ModelFormatOBJ, ModelFormat3DS, ModelFormatVRML, ModelFormatPLY, ModelFormatCOLLADA, ModelFormatU3D, ModelFormatPDF, ModelFormatDXF, ModelFormatFBX, ModelFormatKMZ, ModelFormatCTM, ModelFormatSTL, ModelFormatDXF_3DF, ModelFormatTLS, ModelFormatABC, ModelFormatOSGB, ModelFormatOSGT, ModelFormatGLTF, ModelFormatX3D, ModelFormatLandXML]

class Metashape.**ModelGroup**

ModelGroup objects define groups of multiple models. The grouping is established by assignment of a ModelGroup instance to the Model.group attribute of participating models.

key

Model group identifier.

Type
int

label

Model group label.

Type
str

meta

Model group meta data.

Type
Metashape.MetaData

selected

Current selection state.

Type
bool

class Metashape.**NetworkClient**

NetworkClient class provides access to the network processing server and allows to create and manage tasks.

The following example connects to the server and lists active tasks:

```
>>> import Metashape
>>> client = Metashape.NetworkClient()
>>> client.connect('127.0.0.1')
>>> client.batchList()
```

abortBatch(*batch_id*)

Abort batch.

Parameters
batch_id (*int*) – Batch id.

abortWorker(*worker_id*)

Abort worker.

Parameters
worker_id (*int*) – Worker id.

batchInfo(*batch_id*, *revision=0*)

Get batch information.

Parameters

- **batch_id** (*int*) – Batch id.
- **revision** (*int*) – First revision to get.

Returns
Batch information.

Return type
dict

`batchList`(*revision=0*)

Get list of batches.

Parameters
revision (*int*) – First revision to get.

Returns
List of batches.

Return type
dict

`connect`(*host*, *port=5840*)

Connect to the server.

Parameters

- **host** (*str*) – Server hostname.
- **port** (*int*) – Communication port.

`createBatch`(*path*, *tasks*[, *meta*])

Create new batch.

Parameters

- **path** (*str*) – Project path relative to root folder.
- **tasks** (*list[Metashape.NetworkTask]*) – List of processing tasks to execute.
- **meta** (Metashape.MetaData) – Batch metadata.

Returns
Batch id.

Return type
int

`disconnect`()

Disconnect from the server.

`exportBatches`([*batch_ids*])

Export current state of batches.

Parameters
batch_ids (*list[int]*) – List of batch ids to export.

Returns
Batches data.

Return type
str

`findBatch`(*path*)

Get batch id based on project path.

Parameters
path (*str*) – Project path relative to root folder.

Returns
Batch id.

Return type
int

importBatches(*data*)

Import batches from exported data.

Parameters
data (*str*) – Batches data.

quitWorker(*worker_id*)

Quit worker.

Parameters
worker_id (*int*) – Worker id.

serverInfo(*revision=0*)

Get server information.

Parameters
revision (*int*) – First revision to get.

Returns
Server information.

Return type
dict

serverVersion()

Get server version.

Returns
Server version.

Return type
dict

setBatchPaused(*batch_id*, *paused=True*)

Set batch paused state.

Parameters

- **batch_id** (*int*) – Batch id.
- **paused** (*bool*) – Paused state.

setBatchPriority(*batch_id*, *priority*)

Set batch priority.

Parameters

- **batch_id** (*int*) – Batch id.
- **priority** (*int*) – Batch priority (2 - Highest, 1 - High, 0 - Normal, -1 - Low, -2 - Lowest).

setBatchWorkerLimit(*batch_id*, *worker_limit*)

Set worker limit of the batch.

Parameters

- **batch_id** (*int*) – Batch id.
- **worker_limit** (*int*) – Worker limit of the batch (0 - unlimited).

setMasterServer([*host*])

Set or reset master server.

Parameters
`host` (*str*) – Master server hostname.

setWorkerCapability(*worker_id*, *capability*)

Set worker capability.

Parameters

- `worker_id` (*int*) – Worker id.
- `capability` (*int*) – Worker capability (1 - CPU, 2 - GPU, 3 - Any).

setWorkerCpuEnabled(*worker_id*, *cpu_enabled*)

Set worker CPU enabled flag.

Parameters

- `worker_id` (*int*) – Worker id.
- `cpu_enabled` (*bool*) – CPU enabled flag.

setWorkerGpuMask(*worker_id*, *gpu_mask*)

Set worker GPU mask.

Parameters

- `worker_id` (*int*) – Worker id.
- `gpu_mask` (*int*) – GPU device mask.

setWorkerPaused(*worker_id*, *paused=True*)

Set worker paused state.

Parameters

- `worker_id` (*int*) – Worker id.
- `paused` (*bool*) – Paused state.

setWorkerPriority(*worker_id*, *priority*)

Set worker priority.

Parameters

- `worker_id` (*int*) – Worker id.
- `priority` (*int*) – Worker priority (2 - Highest, 1 - High, 0 - Normal, -1 - Low, -2 - Lowest).

workerInfo(*worker_id*, *revision=0*)

Get worker information.

Parameters

- `worker_id` (*int*) – Worker id.
- `revision` (*int*) – First revision to get.

Returns
Worker information.

Return type
dict

workerList(*revision=0*)

Get list of workers.

Parameters
revision (*int*) – First revision to get.

Returns
List of workers.

Return type
dict

class Metashape.**NetworkTask**

NetworkTask class contains information about network task and its parameters.

The following example creates a new processing task and submits it to the server:

```
>>> import Metashape
>>> task = Metashape.NetworkTask()
>>> task.name = 'MatchPhotos'
>>> task.params['keypoint_limit'] = 40000
>>> client = Metashape.NetworkClient()
>>> client.connect('127.0.0.1')
>>> batch_id = client.createBatch('processing/project.psx', [task])
>>> client.setBatchPaused(batch_id, false)
```

chunks

List of chunks.

Type
list

encode()

Create a dictionary with task parameters.

frames

List of frames.

Type
list

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

params

Task parameters.

Type
dict

class Metashape.OrthoProjection

Orthographic projection.

class Type

Projection type in [Planar, Cylindrical]

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.OrthoProjection

crs

Base coordinate system.

Type
Metashape.CoordinateSystem

matrix

Ortho transformation matrix.

Type
Metashape.Matrix

radius

Cylindrical projection radius.

Type
float

transform(*point*, *source*, *target*)

Transform point coordinates between coordinate systems.

Parameters

- **point** (`Metashape.Vector`) – 2D or 3D point coordinates.
- **source** (`Metashape.OrthoProjection` | `Metashape.CoordinateSystem`) – Source coordinate system.
- **target** (`Metashape.OrthoProjection` | `Metashape.CoordinateSystem`) – Target coordinate system.

Returns
Transformed point coordinates.

Return type
Metashape.Vector

type

Projection type.

Type
Metashape.OrthoProjection.Type

class Metashape.Orthomosaic

Orthomosaic data.

The following sample assigns to the first shape in the chunk the image from the first camera for the orthomosaic patch and updates the mosaic:

```
>>> import Metashape
>>> chunk = Metashape.app.document.chunk
>>> ortho = chunk.orthomosaic
>>> camera = chunk.cameras[0]
>>> shape = chunk.shapes[0]
>>> patch = Metashape.Orthomosaic.Patch()
>>> patch.image_keys = [camera.key]
>>> ortho.patches[shape] = patch
>>> ortho.update()
```

class Patch

Orthomosaic patch.

copy()

Returns a copy of the patch.

Returns
Copy of the patch.

Return type
Metashape.Orthomosaic.Patch

excluded

Excluded flag.

Type
bool

image_keys

Image keys.

Type
list[int]

class Patches

A set of orthomosaic patches.

items()

List of items.

keys()

List of item keys.

values()

List of item values.

bands

List of color bands.

Type
list[str]

bottom

Y coordinate of the bottom side.

Type
float

clear()

Clears orthomosaic data.

copy()

Create a copy of the orthomosaic.

Returns
Copy of the orthomosaic.

Return type
Metashape.Orthomosaic

crs

Coordinate system of orthomosaic.

Type
Metashape.CoordinateSystem

data_type

Data type used to store color values.

Type
Metashape.DataType

height

Orthomosaic height.

Type
int

key

Orthomosaic identifier.

Type
int

label

Orthomosaic label.

Type
str

left

X coordinate of the left side.

Type
float

meta

Orthomosaic meta data.

Type
Metashape.MetaData

modified

Modified flag.

Type
bool

patches

Orthomosaic patches.

Type
Metashape.Orthomosaic.Patches

projection
Orthomosaic projection.

Type
Metashape.OrthoProjection

removeOrthophotos()
Remove orthorectified images from orthomosaic.

renderPreview(*width = 2048*, *height = 2048*[, *progress*])
Generate orthomosaic preview image.

Parameters

- **width** (*int*) – Preview image width.
- **height** (*int*) – Preview image height.
- **progress** (*Callable[[float], None]*) – Progress callback.

Returns
Preview image.

Return type
Metashape.Image

reset([*progress*])
Reset all edits to orthomosaic.

Parameters
progress (*Callable[[float], None]*) – Progress callback.

resolution
Orthomosaic resolution in meters.

Type
float

right
X coordinate of the right side.

Type
float

top
Y coordinate of the top side.

Type
float

update([*progress*])
Apply edits to orthomosaic.

Parameters
progress (*Callable[[float], None]*) – Progress callback.

width
Orthomosaic width.

Type
int

class Metashape.**Photo**

Photo instance

alpha()

Returns alpha channel data.

Returns
Alpha channel data.

Return type
Metashape.Image

copy()

Returns a copy of the photo.

Returns
Copy of the photo.

Return type
Metashape.Photo

image([*channels*][, *datatype*])

Returns image data.

Parameters

- **datatype** (*str*) – pixel data type in ['U8', 'U16', 'U32', 'F16', 'F32', 'F64']
- **channels** (*str*) – color channels to be loaded, e.g. 'RGB', 'RGBA', etc.

Returns
Image data.

Return type
Metashape.Image

imageMeta()

Returns image meta data.

Returns
Image meta data.

Return type
Metashape.MetaData

layer

Layer index in the image file.

Type
int

meta

Frame meta data.

Type
Metashape.MetaData

open(*path*, *layer=0*)

Loads specified image file.

Parameters

- **path** (*str*) – Path to the image file to be loaded.

- **layer** (*int*) – Layer index in case of multipage files.

path

Path to the image file.

Type
str

thumbnail(*width=192*, *height=192*)

Creates new thumbnail with specified dimensions.

Returns
Thumbnail data.

Return type
Metashape.Thumbnail

class Metashape.PointClass

Point class in [Created, Unclassified, Ground, LowVegetation, MediumVegetation, HighVegetation, Building, LowPoint, ModelKeyPoint, Water, Rail, RoadSurface, OverlapPoints, WireGuard, WireConductor, TransmissionTower, WireConnector, BridgeDeck, HighNoise, Car, Manmade]

class Metashape.PointCloud

Point cloud data.

assignClass(*target=0*[, *source*][, *progress*])

Assign class to points.

Parameters

- **target** (Metashape.PointClass) – Target class.
- **source** (Metashape.PointClass | *list*[Metashape.PointClass]) – Classes of points to be replaced.
- **progress** (*Callable[[float], None]*) – Progress callback.

assignClassToSelection(*target=0*[, *source*][, *progress*])

Assign class to selected points.

Parameters

- **target** (Metashape.PointClass) – Target class.
- **source** (Metashape.PointClass | *list*[Metashape.PointClass]) – Classes of points to be replaced.
- **progress** (*Callable[[float], None]*) – Progress callback.

bands

List of color bands.

Type
list[str]

classifyGroundPoints(*max_angle=10.0*, *max_distance=1.0*, *max_terrain_slope=10.0*, *cell_size=50.0*, *erosion_radius=0.0*[, *source_class*][, *return_number*], *keep_existing=False*[, *progress*])

Classify points into ground and non ground classes.

Parameters

- **max_angle** (*float*) – Maximum angle (degrees).

- **max_distance** (*float*) – Maximum distance (meters).
- **max_terrain_slope** (*float*) – Maximum terrain slope angle (degrees).
- **cell_size** (*float*) – Cell size (meters).
- **erosion_radius** (*float*) – Erosion radius (meters).
- **source_class** (Metashape.PointClass) – Class of points to be re-classified.
- **return_number** (*int*) – Point return number to use (0 - any return, 1 - first return, -1 - last return).
- **keep_existing** (*bool*) – Keep existing ground points.
- **progress** (*Callable[[float], None]*) – Progress callback.

classifyPoints([*source*][, *target*], *confidence=0.0*[, *progress*])

Multiclass classification of points.

Parameters

- **source** (Metashape.PointClass) – Class of points to be re-classified.
- **target** (*list[*Metashape.PointClass*]*) – Target point classes for classification.
- **confidence** (*float*) – Required confidence level from 0.0 to 1.0.
- **progress** (*Callable[[float], None]*) – Progress callback.

clear()

Clears point cloud data.

compactPoints([*progress*])

Permanently removes deleted points from point cloud.

Parameters
progress (*Callable[[float], None]*) – Progress callback.

component

Point cloud component.

Type
Metashape.Component

copy()

Create a copy of the point cloud.

Returns
Copy of the point cloud.

Return type
Metashape.PointCloud

cropSelectedPoints([*point_classes*][, *progress*])

Crop selected points.

Parameters

- **point_classes** (Metashape.PointClass | *list[*Metashape.PointClass*]*) – Classes of points to be removed.
- **progress** (*Callable[[float], None]*) – Progress callback.

crs

Reference coordinate system.

Type
Metashape.CoordinateSystem | None

data_type

Data type used to store color values.

Type
Metashape.DataType

enabled

Enables/disables the point cloud.

Type
bool

group

Point cloud group.

Type
Metashape.PointCloudGroup

is_laser_scan

Use point cloud as laser scan.

Type
bool

key

Point cloud identifier.

Type
int

label

Point cloud label.

Type
str

meta

Point cloud meta data.

Type
Metashape.MetaData

modified

Modified flag.

Type
bool

pickPoint(*origin*, *target*, *endpoints=1*)

Returns ray intersection with the point cloud (point on the ray nearest to some point).

Parameters

- **origin** (`Metashape.Vector`) – Ray origin.
- **target** (`Metashape.Vector`) – Point on the ray.

- **endpoints** (*int*) – Number of endpoints to check for (0 - line, 1 - ray, 2 - segment).

Returns
Coordinates of the intersection point.

Return type
Metashape.Vector

point_count
Number of points in point cloud.

Type
int

removePoints(*point_classes*[, *progress*])
Remove points.

Parameters

- **point_classes** (Metashape.PointClass | *list*[Metashape.PointClass]) – Classes of points to be removed.
- **progress** (*Callable[[float], None]*) – Progress callback.

removeSelectedPoints([*point_classes*][, *progress*])
Remove selected points.

Parameters

- **point_classes** (Metashape.PointClass | *list*[Metashape.PointClass]) – Classes of points to be removed.
- **progress** (*Callable[[float], None]*) – Progress callback.

renderDepth(*transform*, *calibration*, *point_size=1*, *resolution=1*, *cull_points=False*, *add_alpha=True*)
Render point cloud depth image for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **point_size** (*int*) – Point size.
- **resolution** (*float*) – Level of detail resolution in screen pixels.
- **cull_points** (*bool*) – Enable normal based culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.

Returns
Rendered image.

Return type
Metashape.Image

renderImage(*transform*, *calibration*, *point_size=1*, *resolution=1*, *cull_points=False*, *add_alpha=True*, *raster_transform=RasterTransformNone*)
Render point cloud image for specified viewpoint.

Parameters

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

- **point_size** (*int*) – Point size.
- **resolution** (*float*) – Level of detail resolution in screen pixels.
- **cull_points** (*bool*) – Enable normal based culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.
- **raster_transform** (Metashape.RasterTransformType) – Raster band transformation.

Returns
Rendered image.

Return type
Metashape.Image

renderMask(*transform*, *calibration*, *point_size=1*, *resolution=1*, *cull_points=False*)

Render point cloud mask image for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **point_size** (*int*) – Point size.
- **resolution** (*float*) – Level of detail resolution in screen pixels.
- **cull_points** (*bool*) – Enable normal based culling.

Returns
Rendered image.

Return type
Metashape.Image

renderNormalMap(*transform*, *calibration*, *point_size=1*, *resolution=1*, *cull_points=False*, *add_alpha=True*)

Render image with point cloud normals for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **point_size** (*int*) – Point size.
- **resolution** (*float*) – Level of detail resolution in screen pixels.
- **cull_points** (*bool*) – Enable normal based culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.

Returns
Rendered image.

Return type
Metashape.Image

renderPreview(*width = 2048*, *height = 2048*[, *transform*], *point_size=1*[, *progress*])

Generate point cloud preview image.

Parameters

- **width** (*int*) – Preview image width.

- **height** (*int*) – Preview image height.
- **transform** (*Metashape.Matrix*) – 4x4 viewpoint transformation matrix.
- **point_size** (*int*) – Point size.
- **progress** (*Callable[[float], None]*) – Progress callback.

Returns
Preview image.

Return type
Metashape.Image

resetFilters()

Reset filters.

restorePoints([*point_classes*][, *progress*])

Restore deleted points.

Parameters

- **point_classes** (*Metashape.PointClass | list[Metashape.PointClass]*) – Classes of points to be restored.
- **progress** (*Callable[[float], None]*) – Progress callback.

selectMaskedPoints(*cameras*, *softness=4*[, *progress*])

Select points based on image masks.

Parameters

- **cameras** (*list[Metashape.Camera]*) – A list of cameras to use for selection.
- **softness** (*float*) – Mask edge softness.
- **progress** (*Callable[[float], None]*) – Progress callback.

selectPointsByColor(*color*, *tolerance=10*, *channels='RGB'*[, *progress*])

Select points based on point colors.

Parameters

- **color** (*list[int]*) – Color to select.
- **tolerance** (*int*) – Color tolerance.
- **channels** (*str*) – Combination of color channels to compare in ['R', 'G', 'B', 'H', 'S', 'V'].
- **progress** (*Callable[[float], None]*) – Progress callback.

selectPointsByShapes([*shapes*][, *progress*])

Select points based on shapes.

Parameters

- **shapes** (*list[Metashape.Shape]*) – A list of shapes to use for selection (selected shapes if not specified).
- **progress** (*Callable[[float], None]*) – Progress callback.

selected

Selects/deselects the point cloud.

Type
bool

setClassesFilter(*point_classes*)

Set filter by point classes.

Parameters
point_classes (Metashape.PointClass | *list*[Metashape.PointClass]) – List of point classes.

setConfidenceFilter(*min_confidence*, *max_confidence*)

Set filter by confidence.

Parameters

- **min_confidence** (*int*) – Minimum confidence value.
- **max_confidence** (*int*) – Maximum confidence value.

setSelectionFilter()

Set filter by selection.

transform

4x4 point cloud transformation matrix.

Type
Metashape.Matrix

updateStatistics([*progress*])

Updates point cloud statistics.

Parameters
progress (*Callable[[float], None]*) – Progress callback.

class Metashape.**PointCloudFormat**

Point cloud format in [PointCloudFormatNone, PointCloudFormatOBJ, PointCloudFormatPLY, PointCloudFormatXYZ, PointCloudFormatLAS, PointCloudFormatExpe, PointCloudFormatU3D, PointCloudFormatPDF, PointCloudFormatE57, PointCloudFormatOC3, PointCloudFormatPotree, PointCloudFormatLAZ, PointCloudFormatCL3, PointCloudFormatPTS, PointCloudFormatPTX, PointCloudFormatDXF, PointCloudFormatCesium, PointCloudFormatPCD, PointCloudFormatSLPK, PointCloudFormatCOPC]

class Metashape.**PointCloudGroup**

PointCloudGroup objects define groups of multiple laser scans. The grouping is established by assignment of a PointCloudGroup instance to the PointCloud.group attribute of participating laser scans.

crs

Reference coordinate system.

Type
Metashape.CoordinateSystem | None

fixed

Fix relative laser scan positions within the group.

Type
bool

key

Asset group identifier.

Type
int

label
Point cloud group label.

Type
str

meta
Asset group meta data.

Type
Metashape.MetaData

selected
Current selection state.

Type
bool

transform
4x4 asset group transformation matrix.

Type
Metashape.Matrix

class Metashape.Preselection
Image pair preselection in [NoPreselection, GenericPreselection, ReferencePreselection]

class Metashape.RPCModel
Rational polynomial model.

copy()
Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.RPCModel

error(*point*, *proj*)
Returns projection error.

Parameters

- **point** (`Metashape.Vector`) – Coordinates of the point to be projected.
- **proj** (`Metashape.Vector`) – Pixel coordinates of the point.

Returns
2D projection error.

Return type
Metashape.Vector

image_offset
Image coordinate offset.

Type
Metashape.Vector

image_scale

Image coordinate scale.

Type
Metashape.Vector

line_den_coeff

Line denominator.

Type
Metashape.Vector

line_num_coeff

Line numerator.

Type
Metashape.Vector

load(*path*[, *format*])

Load RPC model from file.

Parameters

- **path** (*str*) – Path to RPC model file.
- **format** (*str*) – RPC model file format in ['rpc', 'rpb', 'dimap']. Tiled DIMAP files are not supported.

object_offset

Object coordinate offset.

Type
Metashape.Vector

object_scale

Object coordinate scale.

Type
Metashape.Vector

project(*point*)

Returns projected pixel coordinates of the point.

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

Returns
2D projected point coordinates.

Return type
Metashape.Vector

samp_den_coeff

Sample denominator.

Type
Metashape.Vector

samp_num_coeff

Sample numerator.

Type
Metashape.Vector

save(*path*[, *format*])

Save RPC model to file.

Parameters

- **path** (*str*) – Path to RPC model file.
- **format** (*str*) – RPC model file format in ['rpc', 'rpb'].

unproject(*point*)

Returns direction corresponding to the image point.

Parameters
point (Metashape.Vector) – Pixel coordinates of the point.

Returns
3D vector in the camera coordinate system.

Return type
Metashape.Vector

class Metashape.**RasterFormat**

Raster format in [RasterFormatNone, RasterFormatTiles, RasterFormatKMZ, RasterFormatXYZ, RasterFormatMBTiles, RasterFormatWW, RasterFormatTMS, RasterFormatGeoPackage]

class Metashape.**RasterTransform**

Raster transform definition.

calibrateRange()

Auto detect range based on orthomosaic histogram.

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.RasterTransform

enabled

Enable flag.

Type
bool

false_color

False color channels.

Type
list

formula

Raster calculator expression.

Type
str

interpolation

Interpolation enable flag.

Type
bool

palette
Color palette.

Type
dict

range
Palette mapping range.

Type
tuple

reset()
Reset raster transform.

class Metashape.RasterTransformType
Raster transformation type in [RasterTransformNone, RasterTransformValue, RasterTransformPalette]

class Metashape.ReferenceFormat
Reference format in [ReferenceFormatNone, ReferenceFormatXML, ReferenceFormatTEL, ReferenceFormatCSV, ReferenceFormatMavinci, ReferenceFormatBramor, ReferenceFormatAPM]

class Metashape.ReferenceItems
Reference items in [ReferenceItemsCameras, ReferenceItemsMarkers, ReferenceItemsScalebars]

class Metashape.ReferencePreselectionMode
Reference preselection mode in [ReferencePreselectionSource, ReferencePreselectionEstimated, ReferencePreselectionSequential]

class Metashape.Region
Region parameters

center
Region center coordinates.

Type
Metashape.Vector

copy()
Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.Region

rot
Region rotation matrix.

Type
Metashape.Matrix

size
Region size.

Type
Metashape.Vector

class Metashape.**RotationOrder**

Rotation order in [RotationOrderXYZ, RotationOrderXZY, RotationOrderYXZ, RotationOrderYZX, RotationOrderZXY, RotationOrderZYX]

class Metashape.**Scalebar**

Scale bar instance

class Reference

Scale bar reference data

accuracy

Scale bar length accuracy.

Type
float

distance

Scale bar length.

Type
float

enabled

Enabled flag.

Type
bool

chunk

Chunk the scalebar belongs to.

Type
Metashape.Chunk

frames

Scale bar frames.

Type
list[*Metashape.Scalebar*]

group

Scale bar group.

Type
Metashape.ScalebarGroup

key

Scale bar identifier.

Type
int

label

Scale bar label.

Type
str

meta

Scale bar meta data.

Type
Metashape.MetaData

point0

Start of the scale bar.

Type

Metashape.Marker | *Metashape.Camera*

point1

End of the scale bar.

Type

Metashape.Marker | *Metashape.Camera*

reference

Scale bar reference data.

Type

Metashape.Scalebar.Reference

selected

Selects/deselects the scale bar.

Type

bool

class Metashape.ScalebarGroup

ScalebarGroup objects define groups of multiple scale bars. The grouping is established by assignment of a ScalebarGroup instance to the Scalebar.group attribute of participating scale bars.

label

Scale bar group label.

Type

str

selected

Current selection state.

Type

bool

class Metashape.Sensor

Sensor instance

class Reference

Sensor reference data.

accuracy

Sensor location accuracy.

Type

Metashape.Vector

enabled

Location enabled flag.

Type

bool

location

Sensor coordinates.

Type

Metashape.Vector

location_accuracy

Sensor location accuracy.

Type
Metashape.Vector

location_enabled

Location enabled flag.

Type
bool

rotation

Sensor rotation angles.

Type
Metashape.Vector

rotation_accuracy

Sensor rotation accuracy.

Type
Metashape.Vector

rotation_enabled

Rotation enabled flag.

Type
bool

class Type

Sensor type in [Frame, Fisheye, Spherical, Cylindrical, RPC]

antenna

GPS antenna correction.

Type
Metashape.Antenna

bands

List of color bands.

Type
list[str]

black_level

Black level for each band.

Type
list[float]

calibrateFiducials(*resolution=0.014*)

Fit fiducial coordinates to image measurements.

Parameters
resolution (*float*) – Scanning resolution in mm/pix.

calibration

Adjusted calibration of the photo.

Type
Metashape.Calibration

chunk
Chunk the sensor belongs to.

Type
Metashape.Chunk

data_type
Data type used to store color values.

Type
Metashape.DataType

fiducials
Fiducial marks.

Type
list[*Metashape.Marker*]

film_camera
Film camera flag.

Type
bool

fixed
Fix calibration flag.

Type
bool

fixed_calibration
Fix calibration flag.

Type
bool

fixed_location
Fix location flag.

Type
bool

fixed_params
List of fixed calibration parameters.

Type
list[str]

fixed_rotation
Fix rotation flag.

Type
bool

focal_length
Focal length in mm.

Type
float

height
Image height.

Type
int

key
Sensor identifier.

Type
int

label
Sensor label.

Type
str

layer_index
Sensor layer index.

Type
int

location
Sensor plane location.

Type
Metashape.Vector

location_covariance
Sensor plane location covariance.

Type
Metashape.Matrix

makeMaster()
Make this sensor master in the multi-camera system.

master
Master sensor.

Type
Metashape.Sensor

meta
Sensor meta data.

Type
Metashape.MetaData

normalize_sensitivity
Enable sensitivity normalization.

Type
bool

normalize_to_float
Convert pixel values to floating point after normalization.

Type
bool

photo_params
List of image-variant calibration parameters.

Type
list[str]

pixel_height
Pixel height in mm.

Type
float

pixel_size
Pixel size in mm.

Type
Metashape.Vector

pixel_width
Pixel width in mm.

Type
float

planes
Sensor planes.

Type
list[*Metashape.Sensor*]

reference
Sensor reference data.

Type
Metashape.Sensor.Reference

rolling_shutter
Enable rolling shutter compensation.

Type
Metashape.Shutter.Model

rotation
Sensor plane rotation.

Type
Metashape.Matrix

rotation_covariance
Sensor plane rotation covariance.

Type
Metashape.Matrix

sensitivity
Sensitivity for each band.

Type
list[float]

type

Sensor projection model.

Type
Metashape.Sensor.Type

user_calib

Custom calibration used as initial calibration during photo alignment.

Type
Metashape.Calibration

vignetting

Vignetting for each band.

Type
list[*Metashape.Vignetting*]

width

Image width.

Type
int

class Metashape.ServiceType

Service type in [ServiceSketchfab, ServiceMapbox, Service4DMapper, ServicePointscene, ServiceMelown, ServicePointbox, ServicePicterra, ServiceCesium]

class Metashape.Shape

Shape data.

class BoundaryType

Shape boundary type in [NoBoundary, OuterBoundary, InnerBoundary]

class Vertices

Collection of shape vertices

area()

Return area of the shape on DEM.

Returns
Shape area.

Return type
float

areaFitted()

Return 2D area of the shape projected onto the best fitting plane.

Returns
Shape area.

Return type
float

attributes

Shape attributes.

Type
Metashape.MetaData

boundary_type

Shape boundary type.

Type
Metashape.Shape.BoundaryType

geometry

Shape geometry.

Type
Metashape.Geometry | *Metashape.AttachedGeometry*

group

Shape group.

Type
Metashape.ShapeGroup

is_attached

Attached flag.

Type
bool

key

Shape identifier.

Type
int

label

Shape label.

Type
str

perimeter2D()

Return perimeter of the shape on DEM.

Returns
Shape perimeter.

Return type
float

perimeter3D()

Return perimeter of the shape.

Returns
Shape perimeter.

Return type
float

selected

Selects/deselects the shape.

Type
bool

volume(*level='bestfit'*)

Return volume of the shape measured on DEM above and below best fit, mean level or custom level plane.

Parameters
level (*float*) – Plane level: 'bestfit', 'mean' or custom value.

Returns
Shape volumes.

Return type
dict

class Metashape.ShapeGroup

ShapeGroup objects define groups of multiple shapes. The grouping is established by assignment of a Shape-Group instance to the Shape.group attribute of participating shapes.

color

Shape group color.

Type
tuple[int, int, int, int]

enabled

Enable flag.

Type
bool

key

Shape group identifier.

Type
int

label

Shape group label.

Type
str

meta

Shape group meta data.

Type
Metashape.MetaData

selected

Current selection state.

Type
bool

show_labels

Shape labels visibility flag.

Type
bool

class Metashape.Shapes

A set of shapes for a chunk frame.

`addGroup()`

Add new shape group to the set of shapes.

Returns
Created shape group.

Return type
Metashape.ShapeGroup

`addShape()`

Add new shape to the set of shapes.

Returns
Created shape.

Return type
Metashape.Shape

`crs`

Shapes coordinate system.

Type
Metashape.CoordinateSystem

`group`

Default shape group.

Type
Metashape.ShapeGroup

`groups`

List of shape groups.

Type
list[*Metashape.ShapeGroup*]

`items()`

List of items.

`meta`

Shapes meta data.

Type
Metashape.MetaData

`modified`

Modified flag.

Type
bool

`projection`

Shapes projection.

Type
Metashape.OrthoProjection

`remove(`*items*`)`

Remove items from the shape layer.

Parameters
`items` (*list[*`Metashape.Shape` *|* `Metashape.ShapeGroup`*]*) – A list of items to be removed.

shapes
List of shapes.

Type
list[*Metashape.Shape*]

updateAltitudes(*items*[, *progress*])
Update altitudes for items.

Parameters
- **items** (*list[*`Metashape.Shape` *|* `Metashape.ShapeGroup`*]*) – A list of items to be updated.
- **progress** (*Callable[[float], None]*) – Progress callback.

class Metashape.ShapesFormat
Shapes format in [ShapesFormatNone, ShapesFormatSHP, ShapesFormatKML, ShapesFormatDXF, ShapesFormatGeoJSON, ShapesFormatGeoPackage, ShapesFormatCSV]

class Metashape.Shutter
Shutter object contains estimated parameters of the rolling shutter correction model.

class Model
Rolling shutter model in [Disabled, Regularized, Full]

copy()
Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.Shutter

rotation
Rotation matrix of the rolling shutter model.

Type
Metashape.Matrix

translation
Translation vector of the rolling shutter model.

Type
Metashape.Vector

class Metashape.SurfaceType
Surface type in [Arbitrary, HeightField]

class Metashape.Target
Target parameters

code
Target code.

Type
int

coord

Target location.

Type
Metashape.Vector

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.Target

radius

Target radius.

Type
float

class Metashape.TargetType

Target type in [CircularTarget12bit, CircularTarget14bit, CircularTarget16bit, CircularTarget20bit, CircularTarget, CrossTarget]

class Metashape.Tasks

Task classes.

class AddFrames

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

chunk

Chunk to copy frames from.

Type
int

copy_depth_maps

Copy depth maps.

Type
bool

copy_elevation

Copy DEM.

Type
bool

copy_model

Copy model.

Type
bool

copy_orthomosaic
Copy orthomosaic.
Type
bool

copy_point_cloud
Copy point cloud.
Type
bool

copy_tiled_model
Copy tiled model.
Type
bool

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

encode()
Create a dictionary with task parameters.

encodeJSON()
Create a JSON string with task parameters.

frames
List of frame keys to copy.
Type
list[int]

gpu_support
GPU support flag.
Type
bool

name
Task name.
Type
str

target
Task target.
Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.
Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count
Work item count.
Type
int

class AddPhotos

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

filegroups

List of file groups.

Type

list[int]

filenames

List of files to add.

Type

list[str]

gpu_support

GPU support flag.

Type

bool

group

Camera group key.

Type

int

layout

Image layout.

Type

Metashape.ImageLayout

load_reference

Load reference coordinates.

Type

bool

load_rpc_txt

Load satellite RPC data from auxiliary TXT files.

Type

bool

load_xmp_accuracy

Load accuracy from XMP meta data.

Type
bool

load_xmp_antenna

Load GPS/INS offset from XMP meta data.

Type
bool

load_xmp_calibration

Load calibration from XMP meta data.

Type
bool

load_xmp_orientation

Load orientation from XMP meta data.

Type
bool

name

Task name.

Type
str

strip_extensions

Strip file extensions from camera labels.

Type
bool

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class AlignCameras

Task class containing processing parameters.

adaptive_fitting

Enable adaptive fitting of distortion coefficients.

Type
bool

apply(*object* [, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras

List of cameras to align.

Type
list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

min_image

Minimum number of point projections.

Type
int

name

Task name.

Type
str

point_clouds

List of point clouds to align.

Type
list[int]

reset_alignment

Reset current alignment.

Type
bool

subdivide_task

Enable fine-level task subdivision.

Type
bool

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to *Metashape.NetworkTask* to be applied to specified objects.

Parameters

objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class AlignChunks

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

chunks

List of chunks to be aligned.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

downscale

Alignment accuracy (0 - Highest, 1 - High, 2 - Medium, 4 - Low, 8 - Lowest).

Type

int

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

filter_mask

Filter points by mask.

Type

bool

fit_scale

Fit chunk scale during alignment.

Type

bool

generic_preselection

Enables image pair preselection.

Type
bool

gpu_support
GPU support flag.
Type
bool

keypoint_limit
Maximum number of points for each photo.
Type
int

markers
List of markers to be used for marker based alignment.
Type
list[int]

mask_tiepoints
Apply mask filter to tie points.
Type
bool

method
Alignment method (0 - point based, 1 - marker based, 2 - camera based).
Type
int

name
Task name.
Type
str

reference
Chunk to be used as a reference.
Type
int

target
Task target.
Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.
Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count
Work item count.
Type
int

class AnalyzeImages
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras

List of cameras to be analyzed.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

filter_mask

Constrain analyzed image region by mask.

Type

bool

gpu_support

GPU support flag.

Type

bool

name

Task name.

Type

str

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to *Metashape.NetworkTask* to be applied to specified objects.

Parameters

objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class BuildContours

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type

bool

interval

Contour interval.

Type

float

max_value

Maximum value of contour range.

Type

float

min_value

Minimum value of contour range.

Type

float

name

Task name.

Type

str

prevent_intersections

Prevent contour intersections.

Type

bool

source_data

Source data for contour generation.

Type

Metashape.DataSource

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class BuildDem

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

classes

List of point classes to be used for surface extraction.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

flip_x

Flip X axis direction.

Type

bool

flip_y

Flip Y axis direction.

Type

bool

flip_z

Flip Z axis direction.

Type
bool

`frames`
List of frames to process.
Type
list[int]

`gpu_support`
GPU support flag.
Type
bool

`interpolation`
Interpolation mode.
Type
Metashape.Interpolation

`max_workgroup_size`
Maximum workgroup size.
Type
int

`name`
Task name.
Type
str

`projection`
Output projection.
Type
Metashape.OrthoProjection

`region`
Region to be processed.
Type
Metashape.BBox

`replace_asset`
Replace default asset with generated DEM.
Type
bool

`resolution`
Output resolution in meters.
Type
float

`source_data`
Selects between point cloud and tie points.
Type
Metashape.DataSource

`subdivide_task`
Enable fine-level task subdivision.
Type
bool

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

workitem_size_tiles

Number of tiles in a workitem.

Type

int

class BuildDepthMaps

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras

List of cameras to process.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

downscale

Depth map quality (1 - Ultra high, 2 - High, 4 - Medium, 8 - Low, 16 - Lowest).

Type

int

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

filter_mode

Depth map filtering mode.

Type
Metashape.FilterMode

gpu_support

GPU support flag.

Type
bool

max_neighbors

Maximum number of neighbor images to use for depth map generation.

Type
int

max_workgroup_size

Maximum workgroup size.

Type
int

name

Task name.

Type
str

reuse_depth

Enable reuse depth maps option.

Type
bool

subdivide_task

Enable fine-level task subdivision.

Type
bool

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

workitem_size_cameras

Number of cameras in a workitem.

Type
int

class BuildModel

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

blocks_crs

Blocks grid coordinate system.

Type

Metashape.CoordinateSystem

blocks_origin

Blocks grid origin.

Type

Metashape.Vector

blocks_size

Blocks size in coordinate system units.

Type

float

build_texture

Generate preview textures.

Type

bool

cameras

List of cameras to process.

Type

list[int]

classes

List of point classes to be used for surface extraction.

Type

list[int]

clip_to_boundary

Clip to boundary shapes.

Type

bool

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

`export_blocks`
: Export completed blocks.

  Type
  bool

`face_count`
: Target face count.

  Type
  Metashape.FaceCount

`face_count_custom`
: Custom face count.

  Type
  int

`frames`
: List of frames to process.

  Type
  list[int]

`gpu_support`
: GPU support flag.

  Type
  bool

`interpolation`
: Interpolation mode.

  Type
  Metashape.Interpolation

`keep_depth`
: Enable store depth maps option.

  Type
  bool

`max_workgroup_size`
: Maximum workgroup size.

  Type
  int

`name`
: Task name.

  Type
  str

`output_folder`
: Path to output folder.

  Type
  str

`replace_asset`
: Replace default asset with generated model.

  Type
  bool

source_data

Selects between point cloud, tie points, depth maps and laser scans.

Type
Metashape.DataSource

split_in_blocks

Split model in blocks.

Type
bool

subdivide_task

Enable fine-level task subdivision.

Type
bool

surface_type

Type of object to be reconstructed.

Type
Metashape.SurfaceType

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

trimming_radius

Trimming radius (no trimming if zero).

Type
int

vertex_colors

Enable vertex colors calculation.

Type
bool

vertex_confidence

Enable vertex confidence calculation.

Type
bool

volumetric_masks

Enable strict volumetric masking.

Type
bool

workitem_count

Work item count.

Type
int

workitem_size_cameras

Number of cameras in a workitem.

Type
int

class **BuildOrthomosaic**

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

blending_mode

Orthophoto blending mode.

Type
Metashape.BlendingMode

cull_faces

Enable back-face culling.

Type
bool

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

fill_holes

Enable hole filling.

Type
bool

flip_x

Flip X axis direction.

Type
bool

flip_y

Flip Y axis direction.

Type
bool

flip_z

Flip Z axis direction.

Type
bool

frames
List of frames to process.

Type
list[int]

ghosting_filter
Enable ghosting filter.

Type
bool

gpu_support
GPU support flag.

Type
bool

max_workgroup_size
Maximum workgroup size.

Type
int

name
Task name.

Type
str

projection
Output projection.

Type
Metashape.OrthoProjection

refine_seamlines
Refine seamlines based on image content.

Type
bool

region
Region to be processed.

Type
Metashape.BBox

replace_asset
Replace default asset with generated orthomosaic.

Type
bool

resolution
Pixel size in meters.

Type
float

resolution_x
Pixel size in the X dimension in projected units.

Type
float

resolution_y

Pixel size in the Y dimension in projected units.

Type
float

subdivide_task

Enable fine-level task subdivision.

Type
bool

surface_data

Orthorectification surface.

Type
Metashape.DataSource

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

workitem_size_cameras

Number of cameras in a workitem.

Type
int

workitem_size_tiles

Number of tiles in a workitem.

Type
int

class BuildPanorama

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

blending_mode

Panorama blending mode.

Type
Metashape.BlendingMode

camera_groups

List of camera groups to process.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

frames

List of frames to process.

Type

list[int]

ghosting_filter

Enable ghosting filter.

Type

bool

gpu_support

GPU support flag.

Type

bool

height

Height of output panorama.

Type

int

name

Task name.

Type

str

region

Region to be generated.

Type

Metashape.BBox

rotation

Panorama 3x3 orientation matrix.

Type

Metashape.Matrix

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to *Metashape.NetworkTask* to be applied to specified objects.

Parameters

objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

width

Width of output panorama.

Type

int

workitem_count

Work item count.

Type

int

class BuildPointCloud

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

asset

Asset to process.

Type

int

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

frames

List of frames to process.

Type

list[int]

gpu_support

GPU support flag.

Type

bool

keep_depth

Enable store depth maps option.

Type
bool

max_neighbors
Maximum number of neighbor images to use for depth map filtering.
Type
int

max_workgroup_size
Maximum workgroup size.
Type
int

name
Task name.
Type
str

point_colors
Enable point colors calculation.
Type
bool

point_confidence
Enable point confidence calculation.
Type
bool

points_spacing
Desired point spacing (m).
Type
float

replace_asset
Replace default asset with generated point cloud.
Type
bool

source_data
Source data to extract points from.
Type
Metashape.DataSource

subdivide_task
Enable fine-level task subdivision.
Type
bool

target
Task target.
Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

uniform_sampling

Enable uniform point sampling.

Type
bool

workitem_count

Work item count.

Type
int

workitem_size_cameras

Number of cameras in a workitem.

Type
int

class BuildSeamlines

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

epsilon

Contour simplificaion threshold.

Type
float

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class BuildTexture

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

blending_mode

Texture blending mode.

Type

Metashape.BlendingMode

cameras

A list of cameras to be used for texturing.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

fill_holes

Enable hole filling.

Type

bool

ghosting_filter

Enable ghosting filter.

Type
bool

gpu_support
GPU support flag.

Type
bool

max_workgroup_size
Maximum workgroup size (block model only).

Type
int

name
Task name.

Type
str

source_model
Source model.

Type
int

target
Task target.

Type
Metashape.Tasks.TargetType

texture_size
Texture page size.

Type
int

texture_type
Texture type.

Type
Metashape.Model.TextureType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

transfer_texture
Transfer texture.

Type
bool

workitem_count
Work item count.

Type
int

workitem_size_cameras
Number of cameras in a workitem (block model only).

Type
int

`class BuildTiledModel`

Task class containing processing parameters.

`apply(object[, workitem][, progress])`

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk | Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

`classes`

List of point classes to be used for surface extraction.

Type
list[int]

`decode(dict)`

Initialize task parameters with a dictionary.

`decodeJSON(json)`

Initialize task parameters from a JSON string.

`encode()`

Create a dictionary with task parameters.

`encodeJSON()`

Create a JSON string with task parameters.

`face_count`

Number of faces per megapixel of texture resolution.

Type
int

`frames`

List of frames to process.

Type
list[int]

`ghosting_filter`

Enable ghosting filter.

Type
bool

`gpu_support`

GPU support flag.

Type
bool

`keep_depth`

Enable store depth maps option.

Type
bool

max_workgroup_size
Maximum workgroup size.
Type
int

merge
Merge tiled model flag.
Type
bool

name
Task name.
Type
str

operand_asset
Operand asset key.
Type
int

operand_chunk
Operand chunk key.
Type
int

operand_frame
Operand frame key.
Type
int

pixel_size
Target model resolution in meters.
Type
float

replace_asset
Replace default asset with generated tiled model.
Type
bool

source_data
Selects between point cloud and mesh.
Type
Metashape.DataSource

subdivide_task
Enable fine-level task subdivision.
Type
bool

target
Task target.
Type
Metashape.Tasks.TargetType

tile_size

Size of tiles in pixels.

Type
int

toNetworkTask([*objects*])

Convert task to *Metashape.NetworkTask* to be applied to specified objects.

Parameters
objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

transfer_texture

Transfer source model texture to tiled model.

Type
bool

workitem_count

Work item count.

Type
int

workitem_size_cameras

Number of cameras in a workitem.

Type
int

class BuildUV

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

camera

Camera to be used for texturing in CameraMapping mode.

Type
int

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

mapping_mode
Texture mapping mode.
Type
Metashape.MappingMode

name
Task name.
Type
str

page_count
Number of texture pages to generate.
Type
int

pixel_size
Texture resolution in meters.
Type
float

target
Task target.
Type
Metashape.Tasks.TargetType

texture_size
Expected size of texture page at texture generation step.
Type
int

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.
Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

workitem_count
Work item count.
Type
int

class CalculatePointNormals
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.
Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

point_cloud

Point cloud key to process.

Type
int

point_neighbors

Number of point neighbors to use for normal estimation.

Type
int

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class CalibrateCamera

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

border

Border size to ignore.

Type
int

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

fit_b1

Enable optimization of aspect ratio.

Type
bool

fit_b2

Enable optimization of skew coefficient.

Type
bool

fit_cxcy

Enable optimization of principal point coordinates.

Type
bool

fit_f

Enable optimization of focal length coefficient.

Type
bool

fit_k1

Enable optimization of k1 radial distortion coefficient.

Type
bool

fit_k2

Enable optimization of k2 radial distortion coefficient.

Type
bool

fit_k3

Enable optimization of k3 radial distortion coefficient.

Type
bool

fit_k4

Enable optimization of k4 radial distortion coefficient.

Type
bool

fit_p1

Enable optimization of p1 tangential distortion coefficient.

Type
bool

fit_p2

Enable optimization of p2 tangential distortion coefficient.

Type
bool

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class CalibrateColors

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras

List of cameras to process.

Type
list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

source_data

Source data for calibration.

Type
Metashape.DataSource

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

white_balance

Calibrate white balance.

Type
bool

workitem_count

Work item count.

Type
int

class CalibrateReflectance

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

use_reflectance_panels

Use calibrated reflectance panels.

Type
bool

use_sun_sensor

Apply irradiance sensor measurements.

Type
bool

workitem_count

Work item count.

Type
int

class ClassifyGroundPoints

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.

- **progress** (*Callable[[float], None]*) – Progress callback.

cell_size
Cell size (meters).
Type
float

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

encode()
Create a dictionary with task parameters.

encodeJSON()
Create a JSON string with task parameters.

erosion_radius
Erosion radius (meters).
Type
float

gpu_support
GPU support flag.
Type
bool

keep_existing
Keep existing ground points.
Type
bool

max_angle
Maximum angle (degrees).
Type
float

max_distance
Maximum distance (meters).
Type
float

max_terrain_slope
Maximum terrain slope angle (degrees).
Type
float

name
Task name.
Type
str

point_cloud
Point cloud key to classify.
Type
int

return_number

Point return number to use (0 - any return, 1 - first return, -1 - last return).

Type
int

source_class

Class of points to be re-classified.

Type
int

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ClassifyPoints

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

confidence

Required confidence level.

Type
float

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

name
Task name.
Type
str

point_cloud
Point cloud key to classify.
Type
int

source_class
Class of points to be re-classified.
Type
int

subdivide_task
Enable fine-level task subdivision.
Type
bool

target
Task target.
Type
Metashape.Tasks.TargetType

target_classes
Target point classes for classification.
Type
list[int]

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.
Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count
Work item count.
Type
int

class CloseHoles
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.
Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

apply_to_selection
Close holes within selection.

Type
bool

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

level

Hole size threshold in percents.

Type
int

name

Task name.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ColorizeModel

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

model

Key of model to colorize.

Type
int

name

Task name.

Type
str

source_data

Source data to extract colors from.

Type
Metashape.DataSource

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ColorizePointCloud

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.

- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type

bool

max_workgroup_size

Maximum workgroup size.

Type

int

name

Task name.

Type

str

point_cloud

Point cloud key to colorize.

Type

int

source_data

Source data to extract colors from.

Type

Metashape.DataSource

subdivide_task

Enable fine-level task subdivision.

Type

bool

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

workitem_size_cameras

Number of cameras in a workitem.

Type
int

class CompactPointCloud

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

point_cloud

Point cloud key to process.

Type
int

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to *Metashape.NetworkTask* to be applied to specified objects.

Parameters

objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ConvertImages

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk | Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras

List of cameras to process.

Type
list[int]

color_correction

Apply color correction.

Type
bool

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

image_compression

Image compression parameters.

Type
Metashape.ImageCompression

merge_planes

Merge multispectral images.

Type
bool

name

Task name.

Type
str

path

Path to output file.

Type

str

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

update_gps_tags

Update GPS tags.

Type

bool

use_initial_calibration

Transform to initial calibration.

Type

bool

workitem_count

Work item count.

Type

int

class DecimateModel

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

apply_to_selection

Apply to selection.

Type

bool

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

face_count

Target face count.

Type
int

frames

List of frames to process.

Type
list[int]

gpu_support

GPU support flag.

Type
bool

model

Model to process.

Type
int

name

Task name.

Type
str

replace_asset

Replace source model with decimated model.

Type
bool

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class DetectFiducials

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.

- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras

List of cameras to process.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

fiducials_position_corners

Search corners for fiducials.

Type

bool

fiducials_position_sides

Search sides for fiducials.

Type

bool

frame_detector

Detect frame.

Type

bool

frames

List of frames to process.

Type

list[int]

generate_masks

Generate background masks.

Type

bool

generic_detector

Use generic detector.

Type

bool

gpu_support

GPU support flag.

Type

bool

mask_dark_pixels

Mask out dark pixels near frame edge.

Type
bool

name
Task name.
Type
str

right_angle_detector
Use right angle detector.
Type
bool

target
Task target.
Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.
Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

v_shape_detector
Detect V-shape fiducials.
Type
bool

workitem_count
Work item count.
Type
int

class DetectMarkers
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.
Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

cameras
List of cameras to process.
Type
list[int]

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

filter_mask

Ignore masked image regions.

Type
bool

frames

List of frames to process.

Type
list[int]

gpu_support

GPU support flag.

Type
bool

inverted

Detect markers on black background.

Type
bool

maximum_residual

Maximum residual for non-coded targets in pixels.

Type
float

minimum_dist

Minimum distance between targets in pixels (CrossTarget type only).

Type
int

minimum_size

Minimum target radius in pixels to be detected (CrossTarget type only).

Type
int

name

Task name.

Type
str

noparity

Disable parity checking.

Type
bool

target

Task target.

Type
Metashape.Tasks.TargetType

target_type

Type of targets.

Type

Metashape.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

tolerance

Detector tolerance (0 - 100).

Type

int

workitem_count

Work item count.

Type

int

class DetectPowerlines

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type

bool

max_quantization_error

Maximum allowed distance between polyline and smooth continuous curve.

Type

float

min_altitude

Minimum altitude for reconstructed powerlines.

Type
float

n_points_per_line
Maximum number of vertices per detected line.
Type
int

name
Task name.
Type
str

target
Task target.
Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.
Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

use_model
Use model for visibility checks.
Type
bool

workitem_count
Work item count.
Type
int

class DuplicateAsset
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.
Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

asset_key
Asset key.
Type
int

asset_type
Asset type.
Type
Metashape.DataSource

clip_to_boundary
Clip to boundary shapes.

Type
bool

`decode(dict)`

Initialize task parameters with a dictionary.

`decodeJSON(json)`

Initialize task parameters from a JSON string.

`encode()`

Create a dictionary with task parameters.

`encodeJSON()`

Create a JSON string with task parameters.

`gpu_support`

GPU support flag.

Type
bool

`name`

Task name.

Type
str

`target`

Task target.

Type
Metashape.Tasks.TargetType

`toNetworkTask([objects])`

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

`workitem_count`

Work item count.

Type
int

`class DuplicateChunk`

Task class containing processing parameters.

`apply(object[, workitem][, progress])`

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

`chunk`

Chunk to copy.

Type
int

`copy_depth_maps`
Copy depth maps.
Type
bool

`copy_elevations`
Copy DEMs.
Type
bool

`copy_keypoints`
Copy keypoints.
Type
bool

`copy_models`
Copy models.
Type
bool

`copy_orthomosaics`
Copy orthomosaics.
Type
bool

`copy_point_clouds`
Copy point clouds.
Type
bool

`copy_tiled_models`
Copy tiled models.
Type
bool

`decode`(*dict*)
Initialize task parameters with a dictionary.

`decodeJSON`(*json*)
Initialize task parameters from a JSON string.

`encode`()
Create a dictionary with task parameters.

`encodeJSON`()
Create a JSON string with task parameters.

`frames`
List of frame keys to copy.
Type
list[int]

`gpu_support`
GPU support flag.
Type
bool

label

New chunk label.

Type
str

name

Task name.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ExportCameras

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

binary

Enables/disables binary encoding for selected format (if applicable).

Type
bool

bingo_path_geoin

Path to BINGO GEO INPUT file.

Type
str

bingo_path_gps

Path to BINGO GPS/IMU file.

Type
str

bingo_path_image

Path to BINGO IMAGE COORDINATE file.

Type
str

bingo_path_itera
Path to BINGO ITERA file.
Type
str

bingo_save_geoin
Enables/disables export of BINGO GEO INPUT file.
Type
bool

bingo_save_gps
Enables/disables export of BINGO GPS/IMU data.
Type
bool

bingo_save_image
Enables/disables export of BINGO IMAGE COORDINATE file.
Type
bool

bingo_save_itera
Enables/disables export of BINGO ITERA file.
Type
bool

bundler_path_list
Path to Bundler image list file.
Type
str

bundler_save_list
Enables/disables export of Bundler image list file.
Type
bool

chan_rotation_order
Rotation order (CHAN format only).
Type
Metashape.RotationOrder

crs
Output coordinate system.
Type
Metashape.CoordinateSystem

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

encode()
Create a dictionary with task parameters.

encodeJSON()
Create a JSON string with task parameters.

format
: Export format.

Type
Metashape.CamerasFormat

gpu_support
: GPU support flag.

Type
bool

image_orientation
: Image coordinate system (0 - X right, 1 - X up, 2 - X left, 3 - X down).

Type
int

name
: Task name.

Type
str

path
: Path to output file.

Type
str

save_invalid_matches
: Enables/disables export of invalid image matches.

Type
bool

save_markers
: Enables/disables export of manual matching points.

Type
bool

save_points
: Enables/disables export of automatic tie points.

Type
bool

target
: Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
: Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

use_initial_calibration
: Transform image coordinates to initial calibration.

Type
bool

`use_labels`

Enables/disables label based item identifiers.

Type
bool

`workitem_count`

Work item count.

Type
int

`class ExportMarkers`

Task class containing processing parameters.

`apply(object[, workitem][, progress])`

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk | Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

`binary`

Enables/disables binary encoding for selected format (if applicable).

Type
bool

`crs`

Output coordinate system.

Type
Metashape.CoordinateSystem

`decode(dict)`

Initialize task parameters with a dictionary.

`decodeJSON(json)`

Initialize task parameters from a JSON string.

`encode()`

Create a dictionary with task parameters.

`encodeJSON()`

Create a JSON string with task parameters.

`gpu_support`

GPU support flag.

Type
bool

`name`

Task name.

Type
str

`path`

Path to output file.

Type
str

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class ExportMasks

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras

List of cameras to process.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type

bool

name

Task name.

Type

str

path

Path to output file.

Type
str

target
Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count
Work item count.

Type
int

class ExportModel
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

binary
Enables/disables binary encoding (if supported by format).

Type
bool

clip_to_boundary
Clip model to boundary shapes.

Type
bool

colors_rgb_8bit
Convert colors to 8 bit RGB.

Type
bool

comment
Optional comment (if supported by selected format).

Type
str

crs
Output coordinate system.

Type
Metashape.CoordinateSystem

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

embed_texture
Embeds texture inside the model file (if supported by format).
Type
bool

encode()
Create a dictionary with task parameters.

encodeJSON()
Create a JSON string with task parameters.

format
Export format.
Type
Metashape.ModelFormat

gpu_support
GPU support flag.
Type
bool

model
Model key to export.
Type
int

name
Task name.
Type
str

path
Path to output model.
Type
str

precision
Number of digits after the decimal point (for text formats).
Type
int

raster_transform
Raster band transformation.
Type
Metashape.RasterTransformType

save_alpha
Enables/disables alpha channel export.
Type
bool

save_cameras
Enables/disables camera export.

Type
bool

`save_colors`
Enables/disables export of vertex colors.
Type
bool

`save_comment`
Enables/disables comment export.
Type
bool

`save_confidence`
Enables/disables export of vertex confidence.
Type
bool

`save_markers`
Enables/disables marker export.
Type
bool

`save_metadata_xml`
Save metadata.xml file.
Type
bool

`save_normals`
Enables/disables export of vertex normals.
Type
bool

`save_texture`
Enables/disables texture export.
Type
bool

`save_udim`
Enables/disables UDIM texture layout.
Type
bool

`save_uv`
Enables/disables uv coordinates export.
Type
bool

`shift`
Optional shift to be applied to vertex coordinates.
Type
Metashape.Vector

`strip_extensions`
Strips camera label extensions during export.
Type
bool

target

Task target.

Type

Metashape.Tasks.TargetType

texture_format

Texture format.

Type

Metashape.ImageFormat

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

viewpoint

Default view.

Type

Metashape.Viewpoint

workitem_count

Work item count.

Type

int

class ExportOrthophotos

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras

List of cameras to process.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

image_compression
Image compression parameters.
Type
Metashape.ImageCompression

name
Task name.
Type
str

north_up
Use north-up orientation for export.
Type
bool

path
Path to output orthophoto.
Type
str

projection
Output projection.
Type
Metashape.OrthoProjection

raster_transform
Raster band transformation.
Type
Metashape.RasterTransformType

region
Region to be exported.
Type
Metashape.BBox

resolution
Output resolution in meters.
Type
float

resolution_x
Pixel size in the X dimension in projected units.
Type
float

resolution_y
Pixel size in the Y dimension in projected units.
Type
float

save_alpha
Enable alpha channel generation.
Type
bool

save_kml

Enable kml file generation.

Type
bool

save_world

Enable world file generation.

Type
bool

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

white_background

Enable white background.

Type
bool

workitem_count

Work item count.

Type
int

class ExportPointCloud

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

binary

Enables/disables binary encoding for selected format (if applicable).

Type
bool

block_height

Block height in meters.

Type
float

block_width

Block width in meters.

Type
float

classes

List of point classes to be exported.

Type
list[int]

clip_to_boundary

Clip point cloud to boundary shapes.

Type
bool

colors_rgb_8bit

Convert colors to 8 bit RGB.

Type
bool

comment

Optional comment (if supported by selected format).

Type
str

compression

Enable compression (Cesium format only).

Type
bool

crs

Output coordinate system.

Type
Metashape.CoordinateSystem

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

folder_depth

Tileset subdivision depth (Cesium format only).

Type
int

format

Export format.

Type
Metashape.PointCloudFormat

gpu_support

GPU support flag.

Type
bool

image_format
Image data format.
Type
Metashape.ImageFormat

name
Task name.
Type
str

path
Path to output file.
Type
str

point_cloud
Point cloud key to export.
Type
int

raster_transform
Raster band transformation.
Type
Metashape.RasterTransformType

region
Region to be exported.
Type
Metashape.BBox

save_comment
Enable comment export.
Type
bool

save_images
Enable image export.
Type
bool

save_point_classification
Enables/disables export of point classification.
Type
bool

save_point_color
Enables/disables export of point color.
Type
bool

save_point_confidence
Enables/disables export of point confidence.
Type
bool

save_point_index

Enables/disables export of point row and column indices.

Type
bool

save_point_intensity

Enables/disables export of point intensity.

Type
bool

save_point_normal

Enables/disables export of point normal.

Type
bool

save_point_return_number

Enables/disables export of point return number.

Type
bool

save_point_scan_angle

Enables/disables export of point scan angle.

Type
bool

save_point_source_id

Enables/disables export of point source ID.

Type
bool

save_point_timestamp

Enables/disables export of point timestamp.

Type
bool

screen_space_error

Target screen space error (Cesium format only).

Type
float

shift

Optional shift to be applied to point coordinates.

Type
Metashape.Vector

source_data

Selects between point cloud and tie points. If not specified, uses point cloud if available.

Type
Metashape.DataSource

split_in_blocks

Enable tiled export.

Type
bool

subdivide_task

Enable fine-level task subdivision.

Type
bool

target

Task target.

Type
Metashape.Tasks.TargetType

tileset_version

Cesium 3D Tiles format version to export (1.0 or 1.1).

Type
str

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document | Metashape.Chunk | list[Metashape.Chunk]`) – Objects to be processed.

viewpoint

Default view.

Type
Metashape.Viewpoint

workitem_count

Work item count.

Type
int

class ExportRaster

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk | Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

asset

Asset key to export.

Type
int

block_height

Raster block height in pixels.

Type
int

block_width

Raster block width in pixels.

Type
int

clip_to_boundary

Clip raster to boundary shapes.

Type
bool

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

description

Export description.

Type
str

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

format

Export format.

Type
Metashape.RasterFormat

global_profile

Use global profile (GeoPackage format only).

Type
bool

gpu_support

GPU support flag.

Type
bool

height

Raster height.

Type
int

image_compression

Image compression parameters.

Type
Metashape.ImageCompression

image_description

Optional description to be added to image files.

Type
str

image_format

Tile format.

Type
Metashape.ImageFormat

max_zoom_level

Maximum zoom level (GeoPackage, Google Map Tiles, MBTiles and World Wind Tiles formats only).

Type
int

min_zoom_level

Minimum zoom level (GeoPackage, Google Map Tiles, MBTiles and World Wind Tiles formats only).

Type
int

name

Task name.

Type
str

network_links

Enable network links generation for KMZ format.

Type
bool

nodata_value

No-data value (DEM export only).

Type
float

north_up

Use north-up orientation for export.

Type
bool

path

Path to output orthomosaic.

Type
str

projection

Output projection.

Type
Metashape.OrthoProjection

raster_transform

Raster band transformation.

Type
Metashape.RasterTransformType

region

Region to be exported.

Type
Metashape.BBox

resolution

Output resolution in meters.

Type
float

resolution_x

Pixel size in the X dimension in projected units.

Type
float

resolution_y

Pixel size in the Y dimension in projected units.

Type
float

save_alpha

Enable alpha channel generation.

Type
bool

save_kml

Enable kml file generation.

Type
bool

save_scheme

Enable tile scheme files generation.

Type
bool

save_world

Enable world file generation.

Type
bool

source_data

Selects between DEM and orthomosaic.

Type
Metashape.DataSource

split_in_blocks

Split raster in blocks.

Type
bool

target

Task target.

Type
Metashape.Tasks.TargetType

tile_height

Tile height in pixels.

Type
int

tile_width

Tile width in pixels.

Type
int

title

Export title.

Type

str

toNetworkTask([*objects*])

Convert task to *Metashape.NetworkTask* to be applied to specified objects.

Parameters

objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

white_background

Enable white background.

Type

bool

width

Raster width.

Type

int

workitem_count

Work item count.

Type

int

world_transform

2x3 raster-to-world transformation matrix.

Type

Metashape.Matrix

class ExportReference

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

columns

Column order in csv format (n - label, o - enabled flag, x/y/z - coordinates, X/Y/Z - coordinate accuracy, a/b/c - rotation angles, A/B/C - rotation angle accuracy, u/v/w - estimated coordinates, U/V/W - coordinate errors, d/e/f - estimated orientation angles, D/E/F - orientation errors, p/q/r - estimated coordinates variance, i/j/k - estimated orientation angles variance, [] - group of multiple values, | - column separator within group).

Type

str

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

delimiter

Column delimiter in csv format.

Type
str

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

format

Export format.

Type
Metashape.ReferenceFormat

gpu_support

GPU support flag.

Type
bool

items

Items to export in CSV format.

Type
Metashape.ReferenceItems

name

Task name.

Type
str

path

Path to the output file.

Type
str

precision

Number of digits after the decimal point (for CSV format).

Type
int

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ExportReport

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

description

Report description.

Type

str

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

font_size

Font size (pt).

Type

int

gpu_support

GPU support flag.

Type

bool

include_system_info

Include system information.

Type

bool

name

Task name.

Type

str

page_numbers

Enable page numbers.

Type

bool

path

Path to output report.

Type

str

`target`

Task target.

Type

Metashape.Tasks.TargetType

`title`

Report title.

Type

str

`toNetworkTask([objects])`

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

`user_settings`

A list of user defined settings to include on the Processing Parameters page.

Type

list[tuple[str, str]]

`workitem_count`

Work item count.

Type

int

`class ExportShapes`

Task class containing processing parameters.

`apply(object[, workitem][, progress])`

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

`crs`

Output coordinate system.

Type

Metashape.CoordinateSystem

`decode(dict)`

Initialize task parameters with a dictionary.

`decodeJSON(json)`

Initialize task parameters from a JSON string.

`encode()`

Create a dictionary with task parameters.

`encodeJSON()`

Create a JSON string with task parameters.

`format`

Export format.

Type
Metashape.ShapesFormat

`gpu_support`
GPU support flag.
Type
bool

`groups`
A list of shape groups to export.
Type
list[int]

`name`
Task name.
Type
str

`path`
Path to shape file.
Type
str

`polygons_as_polylines`
Save polygons as polylines.
Type
bool

`save_attributes`
Export attributes.
Type
bool

`save_labels`
Export labels.
Type
bool

`save_points`
Export points.
Type
bool

`save_polygons`
Export polygons.
Type
bool

`save_polylines`
Export polylines.
Type
bool

`shift`
Optional shift to be applied to vertex coordinates.
Type
Metashape.Vector

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class ExportTexture

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type

bool

name

Task name.

Type

str

path

Path to output file.

Type

str

raster_transform

Raster band transformation.

Type
Metashape.RasterTransformType

save_alpha
Enable alpha channel export.
Type
bool

target
Task target.
Type
Metashape.Tasks.TargetType

texture_type
Texture type.
Type
Metashape.Model.TextureType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.
Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

workitem_count
Work item count.
Type
int

class ExportTiledModel
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.
Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

clip_to_boundary
Clip tiled model to boundary shapes.
Type
bool

crs
Output coordinate system.
Type
Metashape.CoordinateSystem

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

face_count

Number of faces per megapixel of texture resolution (block model export only).

Type
int

folder_depth

Tileset subdivision depth (Cesium format only).

Type
int

format

Export format.

Type
Metashape.TiledModelFormat

gpu_support

GPU support flag.

Type
bool

image_compression

Image compression parameters.

Type
Metashape.ImageCompression

model_compression

Enable mesh compression (Cesium format only).

Type
bool

model_format

Model format for zip export.

Type
Metashape.ModelFormat

model_group

Block model key to export.

Type
int

name

Task name.

Type
str

path

Path to output model.

Type
str

pixel_size

Target model resolution in meters (block model export only).

Type
float

raster_transform

Raster band transformation.

Type
Metashape.RasterTransformType

screen_space_error

Target screen space error (Cesium format only).

Type
float

target

Task target.

Type
Metashape.Tasks.TargetType

texture_format

Texture format.

Type
Metashape.ImageFormat

tile_size

Size of tiles in pixels (block model export only).

Type
int

tiled_model

Tiled model key to export.

Type
int

tileset_version

Cesium 3D Tiles format version to export (1.0 or 1.1).

Type
str

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

use_tileset_transform

Use tileset transform instead of individual tile transforms (Cesium format only).

Type
bool

workitem_count

Work item count.

Type
int

class FilterPointCloud

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

frames

List of frames to process.

Type

list[int]

gpu_support

GPU support flag.

Type

bool

name

Task name.

Type

str

point_cloud

Point cloud key to filter.

Type

int

point_spacing

Desired point spacing (m).

Type

float

replace_asset

Replace default asset with filtered point cloud.

Type

bool

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class GenerateMasks

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

blur_threshold

Allowed blur radius on a photo in pix (only if mask_defocus=True).

Type

float

cameras

Optional list of cameras to be processed.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

depth_threshold

Maximum depth of masked areas in meters (only if mask_defocus=False).

Type

float

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

fix_coverage

Extend masks to cover whole mesh (only if mask_defocus=True).

Type

bool

gpu_support

GPU support flag.

Type
bool

mask_defocus

Mask defocus areas.

Type
bool

mask_operation

Mask operation.

Type
Metashape.MaskOperation

masking_mode

Mask generation mode.

Type
Metashape.MaskingMode

name

Task name.

Type
str

path

Mask file name template.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

tolerance

Background masking tolerance.

Type
int

workitem_count

Work item count.

Type
int

class GeneratePrescriptionMap

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.

- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

boundary_shape_group

Boundary shape group.

Type
int

breakpoints

Classification breakpoints.

Type
list[float]

cell_size

Step of prescription grid, meters.

Type
float

class_count

Number of classes.

Type
int

classification_method

Index values classification method.

Type
Metashape.ClassificationMethod

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

rates

Fertilizer rate for each class.

Type
list[float]

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ImportCameras

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

crs

Ground coordinate system.

Type
Metashape.CoordinateSystem

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

format

File format.

Type
Metashape.CamerasFormat

gpu_support

GPU support flag.

Type
bool

image_list

Path to image list file (Bundler format only).

Type
str

image_orientation

Image coordinate system (0 - X right, 1 - X up, 2 - X left, 3 - X down).

Type
int

load_image_list

Enable Bundler image list import.

Type
bool

name

Task name.

Type
str

path

Path to the file.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ImportDepthImages

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

color_filenames

List of corresponding color files, if present.

Type
list[str]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

filenames

List of files to import.

Type
list[str]

format

Point cloud format.

Type
Metashape.PointCloudFormat

gpu_support

GPU support flag.

Type
bool

image_path

Path template to output files.

Type
str

multiplane

Import as a multi-camera system

Type
bool

name

Task name.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ImportMarkers

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

path

Path to the file.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to *Metashape.NetworkTask* to be applied to specified objects.

Parameters
objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ImportModel

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

crs

Model coordinate system.

Type
Metashape.CoordinateSystem

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

decode_udim

Load UDIM texture layout.

Type
bool

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

format

Model format.

Type
Metashape.ModelFormat

frame_paths

List of model paths to import in each frame of a multiframe chunk.

Type
list[str]

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

path

Path to model.

Type
str

replace_asset

Replace default asset with imported model.

Type
bool

shift

Optional shift to be applied to vertex coordinates.

Type

Metashape.Vector

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class ImportPointCloud

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

calculate_normals

Calculate point normals.

Type

bool

crs

Point cloud coordinate system.

Type

Metashape.CoordinateSystem

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

format

Point cloud format.

Type
Metashape.PointCloudFormat

`frame_paths`
List of point cloud paths to import in each frame of a multiframe chunk.
Type
list[str]

`gpu_support`
GPU support flag.
Type
bool

`ignore_normals`
Ignore normals in imported file.
Type
bool

`ignore_scanner_origin`
Do not use laser scan origin as scanner position for structured point clouds.
Type
bool

`ignore_trajectory`
Do not attach trajectory to imported point cloud.
Type
bool

`import_images`
Import images embedded in laser scan.
Type
bool

`is_laser_scan`
Import point clouds as laser scans.
Type
bool

`name`
Task name.
Type
str

`path`
Path to point cloud.
Type
str

`point_neighbors`
Number of point neighbors to use for normal estimation.
Type
int

`precision`
Coordinate precision (m). For default precision use 0.
Type
float

replace_asset

Replace default asset with imported point cloud.

Type
bool

scanner_at_origin

Use laser scan origin as scanner position for unstructured point clouds.

Type
bool

shift

Optional shift to be applied to point coordinates.

Type
Metashape.Vector

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

trajectory

Trajectory key to attach.

Type
int

workitem_count

Work item count.

Type
int

class ImportRaster

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

crs

Default coordinate system if not specified in GeoTIFF file.

Type
Metashape.CoordinateSystem

decode(*dict*)

Initialize task parameters with a dictionary.

`decodeJSON`(*json*)

Initialize task parameters from a JSON string.

`encode`()

Create a dictionary with task parameters.

`encodeJSON`()

Create a JSON string with task parameters.

`frames`

List of frames to process.

Type
list[int]

`gpu_support`

GPU support flag.

Type
bool

`has_nodata_value`

No-data value valid flag.

Type
bool

`name`

Task name.

Type
str

`nodata_value`

No-data value.

Type
float

`path`

Path to elevation model in GeoTIFF format.

Type
str

`raster_type`

Type of raster layer to import.

Type
Metashape.DataSource

`replace_asset`

Replace default raster with imported one.

Type
bool

`target`

Task target.

Type
Metashape.Tasks.TargetType

`toNetworkTask`([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ImportReference

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

columns

Column order in csv format (n - label, o - enabled flag, x/y/z - coordinates, X/Y/Z - coordinate accuracy, a/b/c - rotation angles, A/B/C - rotation angle accuracy, [] - group of multiple values, | - column separator within group).

Type
str

create_markers

Create markers for missing entries (csv format only).

Type
bool

crs

Reference data coordinate system (csv format only).

Type
Metashape.CoordinateSystem

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

delimiter

Column delimiter in csv format.

Type
str

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

format

File format.

Type
Metashape.ReferenceFormat

gpu_support

GPU support flag.

Type
bool

group_delimiters

Combine consecutive delimiters in csv format.

Type
bool

ignore_labels

Matches reference data based on coordinates alone (csv format only).

Type
bool

items

List of items to load reference for (csv format only).

Type
Metashape.ReferenceItems

name

Task name.

Type
str

path

Path to the file with reference data.

Type
str

shutter_lag

Shutter lag in seconds (APM format only).

Type
float

skip_rows

Number of rows to skip in (csv format only).

Type
int

target

Task target.

Type
Metashape.Tasks.TargetType

threshold

Error threshold in meters used when ignore_labels is set (csv format only).

Type
float

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count
Work item count.
Type
int

class ImportShapes
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.
Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

boundary_type
Boundary type to be applied to imported shapes.
Type
Metashape.Shape.BoundaryType

columns
Column order in csv format (n - label, x/y/z - coordinates, d - description, [] - group of multiple values, | - column separator within group).
Type
str

crs
Reference data coordinate system (csv format only).
Type
Metashape.CoordinateSystem

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

delimiter
Column delimiter in csv format.
Type
str

encode()
Create a dictionary with task parameters.

encodeJSON()
Create a JSON string with task parameters.

format
Shapes format.
Type
Metashape.ShapesFormat

gpu_support

GPU support flag.

Type
bool

group_delimiters

Combine consequitive delimiters in csv format.

Type
bool

name

Task name.

Type
str

path

Path to shape file.

Type
str

replace

Replace current shapes with new data.

Type
bool

skip_rows

Number of rows to skip in (csv format only).

Type
int

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ImportTiledModel

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

path

Path to tiled model.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ImportTrajectory

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

columns

Column order (t - time, x/y/z - coordinates, a/b/c - rotation angles, space - skip column).

Type
str

crs
Point cloud coordinate system.
Type
Metashape.CoordinateSystem

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

delimiter
CSV delimiter.
Type
str

encode()
Create a dictionary with task parameters.

encodeJSON()
Create a JSON string with task parameters.

format
Trajectory format.
Type
Metashape.TrajectoryFormat

gpu_support
GPU support flag.
Type
bool

name
Task name.
Type
str

path
Trajectory file path.
Type
str

replace_asset
Replace default asset with imported trajectory.
Type
bool

shift
Optional shift to be applied to point coordinates.
Type
Metashape.Vector

skip_rows
Number of rows to skip.

Type
int

`target`
Task target.

Type
Metashape.Tasks.TargetType

`toNetworkTask([objects])`
Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

`workitem_count`
Work item count.

Type
int

`class InvertMasks`
Task class containing processing parameters.

`apply(object[, workitem][, progress])`
Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

`cameras`
List of cameras to process.

Type
list[int]

`decode(dict)`
Initialize task parameters with a dictionary.

`decodeJSON(json)`
Initialize task parameters from a JSON string.

`encode()`
Create a dictionary with task parameters.

`encodeJSON()`
Create a JSON string with task parameters.

`gpu_support`
GPU support flag.

Type
bool

`name`
Task name.

Type
str

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class LoadProject

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

archive

Override project format when using non-standard file extension.

Type

bool

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type

bool

name

Task name.

Type

str

path

Path to project file.

Type
str

read_only
Open project in read only mode.

Type
bool

target
Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to *Metashape.NetworkTask* to be applied to specified objects.

Parameters
objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

workitem_count
Work item count.

Type
int

class MatchPhotos
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.

Parameters
- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras
List of cameras to match.

Type
list[int]

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

downscale
Image alignment accuracy (0 - Highest, 1 - High, 2 - Medium, 4 - Low, 8 - Lowest).

Type
int

downscale_3d
Laser scan alignment accuracy (1 - Highest, 2 - High, 4 - Medium, 8 - Low, 16 - Lowest).

Type
int

`encode()`
: Create a dictionary with task parameters.

`encodeJSON()`
: Create a JSON string with task parameters.

`filter_mask`
: Filter points by mask.

  Type
  bool

`filter_stationary_points`
: Exclude tie points which are stationary across images.

  Type
  bool

`generic_preselection`
: Enable generic preselection.

  Type
  bool

`gpu_support`
: GPU support flag.

  Type
  bool

`guided_matching`
: Enable guided image matching.

  Type
  bool

`keep_keypoints`
: Store keypoints in the project.

  Type
  bool

`keypoint_limit`
: Key point limit.

  Type
  int

`keypoint_limit_3d`
: Key point limit for laser scans.

  Type
  int

`keypoint_limit_per_mpx`
: Key point limit per megapixel.

  Type
  int

`laser_scans_vertical_axis`
: Common laser scans axis.

  Type
  int

mask_tiepoints

Apply mask filter to tie points.

Type
bool

match_laser_scans

Match laser scans using geometric features.

Type
bool

max_workgroup_size

Maximum workgroup size.

Type
int

name

Task name.

Type
str

pairs

User defined list of camera pairs to match.

Type
list[tuple[int, int]]

reference_preselection

Enable reference preselection.

Type
bool

reference_preselection_mode

Reference preselection mode.

Type
Metashape.ReferencePreselectionMode

reset_matches

Reset current matches.

Type
bool

subdivide_task

Enable fine-level task subdivision.

Type
bool

target

Task target.

Type
Metashape.Tasks.TargetType

tiepoint_limit

Tie point limit.

Type
int

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

workitem_size_cameras

Number of cameras in a workitem.

Type

int

workitem_size_pairs

Number of image pairs in a workitem.

Type

int

class MergeAssets

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

assets

List of assets to process.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type

bool

name

Task name.

Type
str

source_data
Asset type.

Type
Metashape.DataSource

target
Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count
Work item count.

Type
int

class MergeChunks
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

chunks
List of chunks to process.

Type
list[int]

copy_depth_maps
Copy depth maps.

Type
bool

copy_elevations
Copy DEMs.

Type
bool

copy_laser_scans
Copy laser scans.

Type
bool

copy_models
Copy models.

Type
bool

`copy_orthomosaics`
Copy orthomosaics.
Type
bool

`copy_point_clouds`
Copy point clouds.
Type
bool

`copy_tiled_models`
Copy tiled models.
Type
bool

`decode`(*dict*)
Initialize task parameters with a dictionary.

`decodeJSON`(*json*)
Initialize task parameters from a JSON string.

`encode`()
Create a dictionary with task parameters.

`encodeJSON`()
Create a JSON string with task parameters.

`gpu_support`
GPU support flag.
Type
bool

`merge_assets`
Merge default assets.
Type
bool

`merge_markers`
Merge markers.
Type
bool

`merge_tiepoints`
Merge tie points.
Type
bool

`name`
Task name.
Type
str

`target`
Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class OptimizeCameras

Task class containing processing parameters.

adaptive_fitting

Enable adaptive fitting of distortion coefficients.

Type
bool

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

fit_b1

Enable optimization of aspect ratio.

Type
bool

fit_b2

Enable optimization of skew coefficient.

Type
bool

fit_corrections

Enable optimization of additional corrections.

Type
bool

fit_cx
Enable optimization of X principal point coordinates.
Type
bool

fit_cy
Enable optimization of Y principal point coordinates.
Type
bool

fit_f
Enable optimization of focal length coefficient.
Type
bool

fit_k1
Enable optimization of k1 radial distortion coefficient.
Type
bool

fit_k2
Enable optimization of k2 radial distortion coefficient.
Type
bool

fit_k3
Enable optimization of k3 radial distortion coefficient.
Type
bool

fit_k4
Enable optimization of k3 radial distortion coefficient.
Type
bool

fit_p1
Enable optimization of p1 tangential distortion coefficient.
Type
bool

fit_p2
Enable optimization of p2 tangential distortion coefficient.
Type
bool

gpu_support
GPU support flag.
Type
bool

name
Task name.
Type
str

target

Task target.

Type

Metashape.Tasks.TargetType

tiepoint_covariance

Estimate tie point covariance matrices.

Type

bool

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class PlanMission

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

attach_viewpoints

Generate additional viewpoints to increase coverage.

Type

bool

capture_distance

Image capture distance (m).

Type

float

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

`group_attached_viewpoints`
Ignore minimum waypoint spacing for additional viewpoints.
Type
bool

`home_point`
Home point shape key.
Type
int

`horizontal_zigzags`
Cover surface with horizontal zigzags instead of vertical.
Type
bool

`interesting_zone`
Interesting zone shape layer key.
Type
int

`max_pitch`
Maximum camera pitch angle.
Type
int

`min_altitude`
Minimum altitude (m).
Type
float

`min_pitch`
Minimum camera pitch angle.
Type
int

`min_waypoint_spacing`
Minimum waypoint spacing (m).
Type
float

`name`
Task name.
Type
str

`overlap`
Overlap percent.
Type
int

`powerlines`
Powerlines shape layer key.
Type
int

restricted_zone

Restricted zone shape layer key.

Type
int

safety_distance

Safety distance (m).

Type
float

safety_zone

Safety zone shape layer key.

Type
int

sensor

Sensor key.

Type
int

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list[*`Metashape.Chunk`*]*) – Objects to be processed.

use_selection

Focus on model selection.

Type
bool

workitem_count

Work item count.

Type
int

class PublishData

Task class containing processing parameters.

account

Account name (Melown service).

Type
str

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

description

Dataset description.

Type
str

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

hostname

Service hostname (4DMapper service).

Type
str

image_compression

Image compression parameters.

Type
Metashape.ImageCompression

is_draft

Mark dataset as draft (Sketchfab service).

Type
bool

is_private

Set dataset access to private (Pointbox and Sketchfab services).

Type
bool

is_protected

Set dataset access to protected (Pointbox service).

Type
bool

max_zoom_level

Maximum zoom level.

Type
int

min_zoom_level

Minimum zoom level.

Type
int

`name`

Task name.

Type

str

`owner`

Account owner (Cesium and Mapbox services).

Type

str

`password`

Account password (4DMapper, Melown, Pointscene and Sketchfab services).

Type

str

`point_classes`

List of point classes to be exported.

Type

list[int]

`projection`

Output projection.

Type

Metashape.CoordinateSystem

`raster_transform`

Raster band transformation.

Type

Metashape.RasterTransformType

`resolution`

Output resolution in meters.

Type

float

`save_camera_track`

Enables/disables export of camera track.

Type

bool

`save_point_color`

Enables/disables export of point colors.

Type

bool

`service`

Service to upload on.

Type

Metashape.ServiceType

`source_data`

Asset type to upload.

Type

Metashape.DataSource

tags

Dataset tags.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

tile_size

Tile size in pixels.

Type
int

title

Dataset title.

Type
str

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

token

Account token (Cesium, Mapbox, Picterra, Pointbox and Sketchfab services).

Type
str

username

Account username (4DMapper, Melown and Pointscene services).

Type
str

workitem_count

Work item count.

Type
int

class ReduceOverlap

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

overlap

Target number of cameras observing each point of the surface.

Type
int

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

use_selection

Focus on model selection.

Type
bool

workitem_count

Work item count.

Type
int

class RefineModel

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (`int`) – Workitem index.
- **progress** (`Callable[[float], None]`) – Progress callback.

cameras
List of cameras to process.
Type
list[int]

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

downscale
Refinement quality (1 - Ultra high, 2 - High, 4 - Medium, 8 - Low, 16 - Lowest).
Type
int

encode()
Create a dictionary with task parameters.

encodeJSON()
Create a JSON string with task parameters.

gpu_support
GPU support flag.
Type
bool

iterations
Number of refinement iterations.
Type
int

name
Task name.
Type
str

smoothness
Smoothing strength. Should be in range [0, 1].
Type
float

target
Task target.
Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.
Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count
Work item count.
Type
int

class RemoveLighting

Task class containing processing parameters.

ambient_occlusion_multiplier

Ambient occlusion multiplier. Should be in range [0.25, 4].

Type
float

ambient_occlusion_path

Path to ambient occlusion texture atlas. Can be empty.

Type
str

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

color_mode

Enable multi-color processing mode.

Type
bool

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

internal_blur

Internal blur. Should be in range [0, 4].

Type
float

mesh_noise_suppression

Mesh normals noise suppression strength. Should be in range [0, 4].

Type
float

name

Task name.

Type
str

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type

int

class RenderDepthMaps

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras

List of cameras to process.

Type

list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type

bool

name

Task name.

Type

str

path_depth

Path to depth map.

Type
str

path_diffuse

Path to diffuse map.

Type
str

path_normals

Path to normal map.

Type
str

save_depth

Enable export of depth map.

Type
bool

save_diffuse

Enable export of diffuse map.

Type
bool

save_normals

Enable export of normal map.

Type
bool

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class ResetMasks

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

cameras

List of cameras to process.

Type
list[int]

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

encode()
Create a dictionary with task parameters.

encodeJSON()
Create a JSON string with task parameters.

gpu_support
GPU support flag.
Type
bool

name
Task name.
Type
str

target
Task target.
Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.
Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count
Work item count.
Type
int

class RunScript
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.
Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

args
Script arguments.
Type
str

code
Script code.
Type
str

decode(*dict*)
Initialize task parameters with a dictionary.

decodeJSON(*json*)
Initialize task parameters from a JSON string.

encode()
Create a dictionary with task parameters.

encodeJSON()
Create a JSON string with task parameters.

gpu_support
GPU support flag.
Type
bool

name
Task name.
Type
str

path
Script path.
Type
str

target
Task target.
Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])
Convert task to `Metashape.NetworkTask` to be applied to specified objects.
Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | `list[Metashape.Chunk]`) – Objects to be processed.

workitem_count
Work item count.
Type
int

class SaveProject
Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])
Apply task to specified object.
Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.

- **progress** (*Callable[[float], None]*) – Progress callback.

archive

Override project format when using non-standard file extension.

Type
bool

chunks

List of chunks to be saved.

Type
list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

name

Task name.

Type
str

path

Path to project.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

version

Project version to save.

Type
str

workitem_count

Work item count.

Type
int

`class SmoothModel`

Task class containing processing parameters.

`apply`(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

`apply_to_selection`

Apply to selected faces.

Type
bool

`decode`(*dict*)

Initialize task parameters with a dictionary.

`decodeJSON`(*json*)

Initialize task parameters from a JSON string.

`encode`()

Create a dictionary with task parameters.

`encodeJSON`()

Create a JSON string with task parameters.

`fix_borders`

Fix borders.

Type
bool

`gpu_support`

GPU support flag.

Type
bool

`model`

Key of model to smooth.

Type
int

`name`

Task name.

Type
str

`preserve_edges`

Preserve edges.

Type
bool

strength

Smoothing strength.

Type
float

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class SmoothPointCloud

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

apply_to_selection

Smooth points within selection.

Type
bool

classes

List of point classes to be smoothed.

Type
list[int]

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

`name`

Task name.

Type
str

`point_cloud`

Key of point cloud to filter.

Type
int

`smoothing_radius`

Desired smoothing radius (m).

Type
float

`target`

Task target.

Type
Metashape.Tasks.TargetType

`toNetworkTask([objects])`

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

`workitem_count`

Work item count.

Type
int

`class TargetType`

Task target type in [DocumentTarget, ChunkTarget, FrameTarget]

`class TrackMarkers`

Task class containing processing parameters.

`apply(object[, workitem][, progress])`

Apply task to specified object.

Parameters
- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

`decode(dict)`

Initialize task parameters with a dictionary.

`decodeJSON(json)`

Initialize task parameters from a JSON string.

`encode()`

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

first_frame

Starting frame index.

Type
int

gpu_support

GPU support flag.

Type
bool

last_frame

Ending frame index.

Type
int

name

Task name.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters
objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

workitem_count

Work item count.

Type
int

class TransformRaster

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters

- **object** (`Metashape.Chunk` | `Metashape.Document`) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

asset

Asset key to transform.

Type
int

`decode(dict)`
Initialize task parameters with a dictionary.

`decodeJSON(json)`
Initialize task parameters from a JSON string.

`encode()`
Create a dictionary with task parameters.

`encodeJSON()`
Create a JSON string with task parameters.

`frames`
List of frames to process.
Type
list[int]

`gpu_support`
GPU support flag.
Type
bool

`height`
Raster height.
Type
int

`name`
Task name.
Type
str

`nodata_value`
No-data value (DEM export only).
Type
float

`north_up`
Use north-up orientation for export.
Type
bool

`operand_asset`
Operand asset key.
Type
int

`operand_chunk`
Operand chunk key.
Type
int

`operand_frame`
Operand frame key.
Type
int

projection

Output projection.

Type

Metashape.OrthoProjection

region

Region to be processed.

Type

Metashape.BBox

replace_asset

Replace default raster with transformed one.

Type

bool

resolution

Output resolution in meters.

Type

float

resolution_x

Pixel size in the X dimension in projected units.

Type

float

resolution_y

Pixel size in the Y dimension in projected units.

Type

float

source_data

Selects between DEM and orthomosaic.

Type

Metashape.DataSource

subtract

Subtraction flag.

Type

bool

target

Task target.

Type

Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to `Metashape.NetworkTask` to be applied to specified objects.

Parameters

objects (`Metashape.Document` | `Metashape.Chunk` | *list*[`Metashape.Chunk`]) – Objects to be processed.

width

Raster width.

Type

int

workitem_count

Work item count.

Type
int

world_transform

2x3 raster-to-world transformation matrix.

Type
Metashape.Matrix

class TriangulateTiePoints

Task class containing processing parameters.

apply(*object*[, *workitem*][, *progress*])

Apply task to specified object.

Parameters
- **object** (Metashape.Chunk | Metashape.Document) – Chunk or Document object to be processed.
- **workitem** (*int*) – Workitem index.
- **progress** (*Callable[[float], None]*) – Progress callback.

decode(*dict*)

Initialize task parameters with a dictionary.

decodeJSON(*json*)

Initialize task parameters from a JSON string.

encode()

Create a dictionary with task parameters.

encodeJSON()

Create a JSON string with task parameters.

gpu_support

GPU support flag.

Type
bool

max_error

Reprojection error threshold.

Type
float

min_image

Minimum number of point projections.

Type
int

name

Task name.

Type
str

target

Task target.

Type
Metashape.Tasks.TargetType

toNetworkTask([*objects*])

Convert task to *Metashape.NetworkTask* to be applied to specified objects.

Parameters

objects (Metashape.Document | Metashape.Chunk | *list*[Metashape.Chunk]) – Objects to be processed.

workitem_count

Work item count.

Type

int

createTask(*name*)

Create task object by its name.

Parameters

name (*str*) – Task name.

Returns

Task object.

Return type

object

class Metashape.**Thumbnail**

Thumbnail instance

copy()

Returns a copy of thumbnail.

Returns

Copy of thumbnail.

Return type

Metashape.Thumbnail

image()

Returns image data.

Returns

Image data.

Return type

Metashape.Image

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

Loads thumbnail from file.

Parameters

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

setImage(*image*)

Parameters

image (Metashape.Image) – Image object with thumbnail data.

class Metashape.**Thumbnails**

A set of thumbnails generated for a chunk frame.

items()

List of items.

keys()

List of item keys.

meta

Thumbnails meta data.

Type

Metashape.MetaData

modified

Modified flag.

Type

bool

values()

List of item values.

class Metashape.**TiePoints**

Tie point cloud instance

class **Cameras**

Collection of `Metashape.TiePoints.Projections` objects indexed by corresponding cameras

class **Filter**

Tie point cloud filter

The following example selects all tie points from the active chunk that have reprojection error higher than defined threshold:

```
>>> chunk = Metashape.app.document.chunk # active chunk
>>> threshold = 0.5
>>> f = Metashape.TiePoints.Filter()
>>> f.init(chunk, criterion = Metashape.TiePoints.Filter.ReprojectionError)
>>> f.selectPoints(threshold)
```

class **Criterion**

Point filtering criterion in [ReprojectionError, ReconstructionUncertainty, ImageCount, ProjectionAccuracy]

init(*points*, *criterion*, *progress*)

Initialize tie points filter based on specified criterion.

Parameters

- **points** (`Metashape.TiePoints | Metashape.Chunk`) – Tie points to filter.
- **criterion** (`Metashape.TiePoints.Filter.Criterion`) – Point filter criterion.
- **progress** (*Callable[[float], None]*) – Progress callback.

max_value

Maximum value.

Type

int | float

min_value

Minimum value.

Type
int | float

removePoints(*threshold*)

Remove points based on specified threshold.

Parameters
threshold (*float*) – Criterion threshold.

resetSelection()

Reset previously made selection.

selectPoints(*threshold*)

Select points based on specified threshold.

Parameters
threshold (*float*) – Criterion threshold.

values

List of values.

Type
list[int] | list[float]

class Point

3D point in the tie point cloud

coord

Point coordinates.

Type
Metashape.Vector

cov

Point coordinates covariance matrix.

Type
Metashape.Matrix

selected

Point selection flag.

Type
bool

track_id

Track index.

Type
int

valid

Point valid flag.

Type
bool

class Points

Collection of 3D points in the tie point cloud

copy()

Returns a copy of points buffer.

Returns
Copy of points buffer.

Return type
Metashape.TiePoints.Points

`resize`(*count*)

Resize points list.

Parameters
`count` (*int*) – new point count

`class Projection`

Projection of the 3D point on the photo

`coord`

2D projection coordinates.

Type
Metashape.Vector

`size`

Point size.

Type
float

`track_id`

Track index.

Type
int

`class Projections`

Collection of `Metashape.TiePoints.Projection` for the camera

`copy`()

Returns a copy of projections buffer.

Returns
Copy of projections buffer.

Return type
Metashape.TiePoints.Projections

`resize`(*count*)

Resize projections list.

Parameters
`count` (*int*) – new projections count

`class Track`

Track in the tie point cloud

`color`

Track color.

Type
tuple[int | float, …]

`class Tracks`

Collection of tracks in the tie point cloud

`copy`()

Returns a copy of tracks buffer.

Returns
Copy of tracks buffer.

Return type
Metashape.TiePoints.Tracks

resize(*count*)
Resize track list.

Parameters
count (*int*) – new track count

bands
List of color bands.

Type
list[str]

cleanup([*progress*])
Remove points with insufficient number of projections.

Parameters
progress (*Callable[[float], None]*) – Progress callback.

copy(*keypoints=True*)
Returns a copy of the tie point cloud.

Parameters
keypoints (*bool*) – copy key points data.

Returns
Copy of the tie point cloud.

Return type
Metashape.TiePoints

cropSelectedPoints()
Crop selected points.

cropSelectedTracks()
Crop selected tie points.

data_type
Data type used to store color values.

Type
Metashape.DataType

export(*path*, *format='obj'*[, *projection*])
Export tie points.

Parameters

- **path** (*str*) – Path to output file.
- **format** (*str*) – Export format in [‘obj’, ‘ply’].
- **projection** (Metashape.Matrix | Metashape.CoordinateSystem) – Sets output projection.

meta
Tie points meta data.

Type
Metashape.MetaData

modified

Modified flag.

Type
bool

pickPoint(*origin*, *target*, *endpoints=1*)

Returns ray intersection with the tie point cloud (point on the ray nearest to some point).

Parameters

- **origin** (Metashape.Vector) – Ray origin.
- **target** (Metashape.Vector) – Point on the ray.
- **endpoints** (*int*) – Number of endpoints to check for (0 - line, 1 - ray, 2 - segment).

Returns
Coordinates of the intersection point.

Return type
Metashape.Vector

points

List of points.

Type
Metashape.TiePoints.Points

projections

Point projections for each photo.

Type
Metashape.TiePoints.Projections

removeKeypoints()

Remove keypoints from tie point cloud.

removeSelectedPoints()

Remove selected points.

removeSelectedTracks()

Remove selected tie points.

renderDepth(*transform*, *calibration*, *point_size=1*, *cull_points=False*, *add_alpha=True*)

Render tie points depth image for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **point_size** (*int*) – Point size.
- **cull_points** (*bool*) – Enable normal based culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.

Returns
Rendered image.

Return type
Metashape.Image

renderImage(*transform*, *calibration*, *point_size=1*, *cull_points=False*, *add_alpha=True*, *raster_transform=RasterTransformNone*)

Render tie points image for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **point_size** (*int*) – Point size.
- **cull_points** (*bool*) – Enable normal based culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.
- **raster_transform** (Metashape.RasterTransformType) – Raster band transformation.

Returns
Rendered image.

Return type
Metashape.Image

renderMask(*transform*, *calibration*, *point_size=1*, *cull_points=False*)

Render tie points mask image for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **point_size** (*int*) – Point size.
- **cull_points** (*bool*) – Enable normal based culling.

Returns
Rendered image.

Return type
Metashape.Image

renderNormalMap(*transform*, *calibration*, *point_size=1*, *cull_points=False*, *add_alpha=True*)

Render image with tie points normals for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **point_size** (*int*) – Point size.
- **cull_points** (*bool*) – Enable normal based culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.

Returns
Rendered image.

Return type
Metashape.Image

renderPreview(*width = 2048*, *height = 2048*[, *transform*], *point_size=1*[, *progress*])

Generate tie points preview image.

Parameters

- **width** (*int*) – Preview image width.
- **height** (*int*) – Preview image height.
- **transform** (Metashape.Matrix) – 4x4 viewpoint transformation matrix.
- **point_size** (*int*) – Point size.
- **progress** (*Callable[[float], None]*) – Progress callback.

Returns
Preview image.

Return type
Metashape.Image

tracks

List of tracks.

Type
Metashape.TiePoints.Tracks

class Metashape.TiledModel

Tiled model data.

class FaceCount

Tiled model face count in [LowFaceCount, MediumFaceCount, HighFaceCount]

bands

List of color bands.

Type
list[str]

clear()

Clears tiled model data.

copy()

Create a copy of the tiled model.

Returns
Copy of the tiled model.

Return type
Metashape.TiledModel

crs

Reference coordinate system.

Type
Metashape.CoordinateSystem | None

data_type

Data type used to store color values.

Type
Metashape.DataType

key
Tiled model identifier.

Type
int

label
Tiled model label.

Type
str

meta
Tiled model meta data.

Type
Metashape.MetaData

modified
Modified flag.

Type
bool

pickPoint(*origin*, *target*, *endpoints=1*)
Returns ray intersection with the tiled model.

Parameters

- **origin** (`Metashape.Vector`) – Ray origin.
- **target** (`Metashape.Vector`) – Point on the ray.
- **endpoints** (*int*) – Number of endpoints to check for (0 - line, 1 - ray, 2 - segment).

Returns
Coordinates of the intersection point.

Return type
Metashape.Vector

renderDepth(*transform*, *calibration*, *resolution=1*, *cull_faces=True*, *add_alpha=True*)
Render tiled model depth image for specified viewpoint.

Parameters

- **transform** (`Metashape.Matrix`) – Camera location.
- **calibration** (`Metashape.Calibration`) – Camera calibration.
- **resolution** (*float*) – Level of detail resolution in screen pixels.
- **cull_faces** (*bool*) – Enable back-face culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.

Returns
Rendered image.

Return type
Metashape.Image

renderImage(*transform*, *calibration*, *resolution=1*, *cull_faces=True*, *add_alpha=True*, *raster_transform=RasterTransformNone*)

Render tiled model image for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **resolution** (*float*) – Level of detail resolution in screen pixels.
- **cull_faces** (*bool*) – Enable back-face culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.
- **raster_transform** (Metashape.RasterTransformType) – Raster band transformation.

Returns
Rendered image.

Return type
Metashape.Image

renderMask(*transform*, *calibration*, *resolution=1*, *cull_faces=True*)

Render tiled model mask image for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **resolution** (*float*) – Level of detail resolution in screen pixels.
- **cull_faces** (*bool*) – Enable back-face culling.

Returns
Rendered image.

Return type
Metashape.Image

renderNormalMap(*transform*, *calibration*, *resolution=1*, *cull_faces=True*, *add_alpha=True*)

Render image with tiled model normals for specified viewpoint.

Parameters

- **transform** (Metashape.Matrix) – Camera location.
- **calibration** (Metashape.Calibration) – Camera calibration.
- **resolution** (*float*) – Level of detail resolution in screen pixels.
- **cull_faces** (*bool*) – Enable back-face culling.
- **add_alpha** (*bool*) – Generate image with alpha channel.

Returns
Rendered image.

Return type
Metashape.Image

renderPreview(*width = 2048*, *height = 2048*[, *transform*][, *progress*])

Generate tiled model preview image.

Parameters

- **width** (*int*) – Preview image width.
- **height** (*int*) – Preview image height.
- **transform** (Metashape.Matrix) – 4x4 viewpoint transformation matrix.
- **progress** (*Callable[[float], None]*) – Progress callback.

Returns
Preview image.

Return type
Metashape.Image

transform

4x4 tiled model transformation matrix.

Type
Metashape.Matrix

class Metashape.**TiledModelFormat**

Tiled model format in [TiledModelFormatNone, TiledModelFormatTLS, TiledModelFormatLOD, TiledModelFormatZIP, TiledModelFormatCesium, TiledModelFormatSLPK, TiledModelFormatOSGB, TiledModelFormatOSGT, TiledModelFormat3MX]

class Metashape.**TrajectoryFormat**

Trajectory format in [TrajectoryFormatNone, TrajectoryFormatCSV, TrajectoryFormatSBET, TrajectoryFormatSOL, TrajectoryFormatTRJ]

class Metashape.**Utils**

Utility functions.

createChessboardImage(*calib*, *cell_size=150*, *max_tilt=30*)

Synthesizes photo of a chessboard.

Parameters

- **calib** (Metashape.Calibration) – Camera calibration.
- **cell_size** (*float*) – Chessboard cell size.
- **max_tilt** (*float*) – Maximum camera tilt in degrees.

Returns
Resulting image.

Return type
Metashape.Image

createDifferenceMask(*image*, *background*, *tolerance=10*, *fit_colors=True*)

Creates mask from a pair of images or an image and specified color.

Parameters

- **image** (Metashape.Image) – Image to be masked.
- **background** (Metashape.Image *| tuple[int, ...]*) – Background image or color value.

- **tolerance** (*int*) – Tolerance value.
- **fit_colors** (*bool*) – Enables white balance correction.

Returns
Resulting mask.

Return type
Metashape.Image

createMarkers(*chunk*, *projections*)

Creates markers from a list of non coded projections.

Parameters

- **chunk** (Metashape.Chunk) – Chunk to create markers in.
- **projections** (*list[tuple[*Metashape.Camera, Metashape.Target*]]*) – List of marker projections.

detectTargets(*image*, *type=TargetCircular12bit*, *tolerance=50*, *inverted=False*, *noparity=False*[, *minimum_size*][, *minimum_dist*])

Detect targets on the image.

Parameters

- **image** (Metashape.Image) – Image to process.
- **type** (Metashape.TargetType) – Type of targets.
- **tolerance** (*int*) – Detector tolerance (0 - 100).
- **inverted** (*bool*) – Detect markers on black background.
- **noparity** (*bool*) – Disable parity checking.
- **minimum_size** (*int*) – Minimum target radius in pixels to be detected (CrossTarget type only).
- **minimum_dist** (*int*) – Minimum distance between targets in pixels (CrossTarget type only).

Returns
List of detected targets.

Return type
list[*Metashape.Target*]

dmat2euler(*R*, *dR*, *euler_angles=EulerAnglesYPR*)

Calculate tangent euler rotation vector from tangent rotation matrix.

Parameters

- **R** (Metashape.Matrix) – Rotation matrix.
- **dR** (Metashape.Matrix) – Tangent rotation matrix.
- **euler_angles** (Metashape.EulerAngles) – Euler angles to use.

Returns
Tangent rotation angles in degrees.

Return type
Metashape.Vector

estimateImageQuality(*image*[, *mask*])

Estimate image sharpness.

Parameters

- **image** (`Metashape.Image`) – Image to be analyzed.
- **mask** (`Metashape.Image`) – Mask of the analyzed image region.

Returns
Quality metric.

Return type
float

euler2mat(*rotation*, *euler_angles=EulerAnglesYPR*)

Calculate camera to world rotation matrix from euler rotation angles.

Parameters

- **rotation** (`Metashape.Vector`) – Rotation vector.
- **euler_angles** (`Metashape.EulerAngles`) – Euler angles to use.

Returns
Rotation matrix.

Return type
Metashape.Matrix

mat2euler(*R*, *euler_angles=EulerAnglesYPR*)

Calculate euler rotation angles from camera to world rotation matrix.

Parameters

- **R** (`Metashape.Matrix`) – Rotation matrix.
- **euler_angles** (`Metashape.EulerAngles`) – Euler angles to use.

Returns
Rotation angles in degrees.

Return type
Metashape.Vector

mat2opk(*R*)

Calculate omega, phi, kappa from camera to world rotation matrix.

Parameters
R (`Metashape.Matrix`) – Rotation matrix.

Returns
Omega, phi, kappa angles in degrees.

Return type
Metashape.Vector

mat2ypr(*R*)

Calculate yaw, pitch, roll from camera to world rotation matrix.

Parameters
R (`Metashape.Matrix`) – Rotation matrix.

Returns
Yaw, pitch roll angles in degrees.

Return type
Metashape.Vector

opk2mat(*angles*)
Calculate camera to world rotation matrix from omega, phi, kappa angles.

Parameters
angles (`Metashape.Vector`) – Omega, phi, kappa angles in degrees.

Returns
Rotation matrix.

Return type
Metashape.Matrix

ypr2mat(*angles*)
Calculate camera to world rotation matrix from yaw, pitch, roll angles.

Parameters
angles (`Metashape.Vector`) – Yaw, pitch, roll angles in degrees.

Returns
Rotation matrix.

Return type
Metashape.Matrix

class Metashape.Vector
n-component vector

```
>>> import Metashape
>>> vect = Metashape.Vector( (1, 2, 3) )
>>> vect2 = vect.copy()
>>> vect2.size = 4
>>> vect2.w = 5
>>> vect2 *= -1.5
>>> vect.size = 4
>>> vect.normalize()
>>> Metashape.app.messageBox("Scalar product is " + str(vect2 * vect))
```

copy()
Return a copy of the vector.

Returns
A copy of the vector.

Return type
Metashape.Vector

cross(*a*, *b*)
Cross product of 2 vectors.

Parameters

- **a** (`Metashape.Vector`) – First vector.
- **b** (`Metashape.Vector`) – Second vector.

Returns
Cross product.

Return type
Metashape.Vector

norm()
Return norm of the vector.

norm2()
Return squared norm of the vector.

normalize()
Normalize vector to the unit length.

normalized()
Return a new, normalized vector.

Returns
a normalized copy of the vector

Return type
Metashape.Vector

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

zero()
Set all elements to zero.

class Metashape.Version
Version object contains application version numbers.

build
Build number.

Type
int

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.Version

major

Major version number.

Type
int

micro

Micro version number.

Type
int

minor

Minor version number.

Type
int

class Metashape.Viewpoint(*app*)

Represents viewpoint in the model view

center

Camera center.

Type
Metashape.Vector

coo

Center of orbit.

Type
Metashape.Vector

copy()

Return a copy of the object.

Returns
A copy of the object.

Return type
Metashape.Viewpoint

fov

Camera vertical field of view in degrees.

Type
float

height

OpenGL window height.

Type

int

mag

Camera magnification defined by distance to the center of rotation.

Type

float

rot

Camera rotation matrix.

Type

Metashape.Matrix

width

OpenGL window width.

Type

int

class Metashape.Vignetting

Vignetting polynomial

copy()

Return a copy of the object.

Returns

A copy of the object.

Return type

Metashape.Vignetting

CHAPTER

THREE

PYTHON API CHANGE LOG

3.1 Metashape version 2.1.0

- Added Component and ModelGroup classes
- Added TrajectoryData, LaserScansData and DepthMapsAndLaserScansData to DataSource enum
- Added PointCloudFormatCOPC to PointCloudFormat enum
- Added ModelViewElevation to ModelView.ModelViewMode enum
- Added TiePointsViewElevation to ModelView.TiePointsViewMode enum
- Added TiledModelViewElevation to ModelView.TiledModelViewMode enum
- Added Chunk.mergeComponents() and Chunk.splitComponents() methods
- Added Elevation.pickPoint() method
- Added ModelView.captureVideo() method
- Added Camera.component attribute
- Added loop and smooth attributes to CameraTrack class
- Added component, components, model_group and model_groups attributes to Chunk class
- Added crs, group and transform attributes to Model class
- Added PointCloud.component attribute
- Added replace_asset and frames attributes to BuildModel, BuildTiledModel, BuildPointCloud, BuildDem, BuildOrthomosaic, DecimateModel, FilterPointCloud, ImportRaster and TransformRaster classes
- Added split_in_blocks, blocks_crs, blocks_size, blocks_origin, clip_to_boundary, export_blocks, build_texture and output_folder attributes to BuildModel class
- Added workitem_size_cameras and max_workgroup_size attributes to BuildTexture class
- Added BuildUV.pixel_size attribute
- Added ClassifyGroundPoints.max_terrain_slope attribute
- Added ExportPointCloud.tileset_version attribute
- Added ExportRaster.asset attribute
- Added model attribute to ColorizeModel and SmoothModel classes
- Added tiled_model, tileset_version, model_group, pixel_size, tile_size and face_count attributes to ExportTiledModel class
- Added replace_asset and frame_paths attributes to ImportModel class

- Added match_laser_scans, downscale_3d, keypoint_limit_3d and laser_scans_vertical_axis attributes to MatchPhotos class
- Added classes and apply_to_selection attributes to SmoothPointCloud class
- Added ImportPointCloud.ignore_normals attribute
- Added replace_asset and frames arguments to Chunk.buildModel(), Chunk.buildTiledModel(), Chunk.buildPointCloud(), Chunk.buildDem(), Chunk.buildOrthomosaic(), Chunk.decimateModel(), Chunk.filterPointCloud(), Chunk.importRaster() and Chunk.transformRaster() methods
- Added replace_asset and frame_paths arguments to Chunk.importModel() method
- Added split_in_blocks, blocks_crs, blocks_size, blocks_origin, clip_to_boundary, export_blocks, build_texture and output_folder arguments to Chunk.buildModel() method
- Added workitem_size_cameras and max_workgroup_size arguments to Chunk.buildTexture() method
- Added pixel_size argument to Chunk.buildUV() method
- Added max_terrain_slope argument to Chunk.classifyGroundPoints() method
- Added tileset_version argument to Chunk.exportPointCloud() method
- Added asset argument to Chunk.exportRaster() method
- Added model argument to Chunk.colorizeModel() and Chunk.smoothModel() methods
- Added tiled_model, tileset_version, model_group, pixel_size, tile_size and face_count arguments to Chunk.exportTiledModel() method
- Added match_laser_scans, downscale_3d, keypoint_limit_3d and laser_scans_vertical_axis arguments to Chunk.matchPhotos() method
- Added classes and apply_to_selection arguments to Chunk.smoothPointCloud() method
- Added ignore_normals argument to Chunk.importPointCloud() method
- Added publish argument to CloudClient.uploadProject() method
- Replaced ExportTiledModel.use_rtc_center attribute with use_tileset_transform
- Replaced use_rtc_center argument in Chunk.exportTiledModel() method with use_tileset_transform
- Renamed RefineMesh class to RefineModel
- Renamed Chunk.refineMesh() method to refineModel()
- Renamed Model.transform() method to transformVertices()
- Renamed NetworkClient.serverInfo() method to serverVersion()
- Renamed NetworkClient.serverStatus() method to serverInfo()
- Renamed NetworkClient.batchStatus() method to batchInfo()
- Renamed NetworkClient.dumpBatches() method to exportBatches()
- Renamed NetworkClient.loadBatches() method to importBatches()
- Renamed NetworkClient.setBatchNodeLimit() method to setBatchWorkerLimit()
- Renamed NetworkClient.nodeList() method to workerList()
- Renamed NetworkClient.nodeStatus() method to workerInfo()
- Renamed NetworkClient.quitNode() method to quitWorker()
- Renamed NetworkClient.abortNode() method to abortWorker()

- Renamed NetworkClient.setNodeCPUEnable() method to setWorkerCpuEnabled()
- Renamed NetworkClient.setNodeCapability() method to setWorkerCapability()
- Renamed NetworkClient.setNodeGPUMask() method to setWorkerGpuMask()
- Renamed NetworkClient.setNodePaused() method to setWorkerPaused()
- Renamed NetworkClient.setNodePriority() method to setWorkerPriority()
- Renamed NetworkTask.supports_gpu attribute to gpu_support
- Renamed supports_gpu attribute to gpu_support in task classes
- Renamed DecimateModel.asset attribute to model
- Renamed TransformRaster.data_source attribute to source_data
- Renamed RenderDepthMaps.export_depth, export_diffuse and export_normals attributes to save_depth, save_diffuse and save_normals
- Renamed asset argument in Chunk.decimateModel() method to model
- Renamed data_source argument in Chunk.transformRaster() method to source_data
- Added .pyi stub file to stand-alone Python module for autocompletion in external IDEs

3.2 Metashape version 2.0.4

- Added borrowLicense() and returnLicense() methods to License class
- Added removeTextures(), removeUV(), removeVertexColors() and removeVertexConfidence() methods to Model class
- Added License.expiration attribute
- Added publish argument to CloudClient.uploadProject() method
- Added format argument to RPCModel.load() and RPCModel.save() methods

3.3 Metashape version 2.0.3

- Added SmoothPointCloud class
- Added Chunk.smoothPointCloud() method
- Added enabled and selected attributes to PointCloud class
- Added mask_dark_pixels and frame_detector attributes to DetectFiducials class
- Added mask_dark_pixels and frame_detector arguments to Chunk.detectFiducials() method

3.4 Metashape version 2.0.2

- Added PointCloudGroup class
- Added TiledModelFormat3MX to TiledModelFormat enum
- Added Chunk.addPointCloudGroup() and Chunk.findPointCloudGroup() methods
- Added Chunk.point_cloud_groups attribute
- Added PointCloud.group and PointCloud.is_laser_scan attributes

3.5 Metashape version 2.0.1

- Added License.install() method
- Added DetectFiducials.v_shape_detector attribute
- Added model and save_metadata_xml attributes to ExportModel task
- Added v_shape_detector argument to Chunk.detectFiducials() method
- Added model and save_metadata_xml arguments to Chunk.exportModel() method
- Replaced license_key argument with activation_params in License.activateOffline() method

3.6 Metashape version 2.0.0

- Added TrajectoryFormat enum
- Added DisplacementMap to Model.TextureType enum
- Added ImportTrajectory class
- Added ImportDepthImages class
- Added Chunk.importTrajectory() method
- Added Chunk.importDepthImages() method
- Added AlignCameras.point_clouds attribute
- Added ImportDepthImages.color_filenames attribute
- Added precision, is_laser_scan, replace_asset, import_images, scanner_at_origin, ignore_scanner_origin, ignore_trajectory, trajectory and frame_paths attributes to ImportPointCloud class
- Added keep_existing, return_number and point_cloud attributes to ClassifyGroundPoints class
- Added point_cloud attribute to ClassifyPoints, ColorizePointCloud, CalculatePointNormals, CompactPointCloud and ExportPointCloud classes
- Added max_quantization_error attribute to DetectPowerlines class
- Added use_rtc_center attribute to ExportTiledModel class
- Added merge_assets, copy_laser_scans, copy_depth_maps, copy_point_clouds, copy_models, copy_tiled_models, copy_elevations and copy_orthomosaics attributes to MergeChunks class
- Added point_clouds argument to Chunk.alignCameras() method
- Added color_filenames argument to Chunk.importDepthImages() method

- Added precision, is_laser_scan, replace_asset, import_images, scanner_at_origin, ignore_scanner_origin, ignore_trajectory, trajectory and frame_paths arguments to Chunk.importPointCloud() method
- Added point_cloud argument to Chunk.calculatePointNormals(), Chunk.colorizePointCloud() and Chunk.exportPointCloud() methods
- Added max_quantization_error argument to Chunk.detectPowerlines() method
- Added keep_existing and return_number arguments to PointCloud.classifyGroundPoints() method
- Added use_rtc_center argument to Chunk.exportTiledModel() method
- Added merge_assets, copy_laser_scans, copy_depth_maps, copy_point_clouds, copy_models, copy_tiled_models, copy_elevations and copy_orthomosaics arguments to Document.mergeChunks() method
- Added drone_name, payload_name and payload_position arguments to CameraTrack.save() method
- Change default source_data argument value for Chunk.buildModel() and Chunk.buildTiledModel() methods to DepthMapsData
- Renamed PointsFormat enum to PointCloudFormat
- Renamed ModelView.PointCloudViewMode enum to ModelView.TiePointsViewMode
- Renamed ModelView.DenseCloudViewMode enum to ModelView.PointCloudViewMode and added PointCloudViewSolid, PointCloudViewIntensity, PointCloudViewElevation, PointCloudViewReturnNumber, PointCloudViewScanAngle, PointCloudViewSourceId enumeration values
- Renamed DataSource.PointCloudData enum value to DataSource.TiePointsData
- Renamed DataSource.DenseCloudData enum value to DataSource.PointCloudData
- Renamed PointCloud class to TiePoints
- Renamed DenseCloud class to PointCloud
- Renamed AnalyzePhotos class to AnalyzeImages
- Renamed BuildDenseCloud class to BuildPointCloud
- Renamed CalibrateLens class to CalibrateCamera
- Renamed ColorizeDenseCloud class to ColorizePointCloud
- Renamed CompactDenseCloud class to CompactPointCloud
- Renamed ExportDepth class to RenderDepthMaps
- Renamed ExportPoints class to ExportPointCloud
- Renamed FilterDenseCloud class to FilterPointCloud
- Renamed ImportPoints class to ImportPointCloud
- Renamed TriangulatePoints class to TriangulateTiePoints
- Renamed Chunk.addDenseCloud() method to addPointCloud()
- Renamed Chunk.analyzePhotos() method to analyzeImages()
- Renamed Chunk.buildDenseCloud() method to buildPointCloud()
- Renamed Chunk.colorizeDenseCloud() method to colorizePointCloud()
- Renamed Chunk.exportPoints() method to exportPointCloud()
- Renamed Chunk.filterDenseCloud() method to filterPointCloud()
- Renamed Chunk.findDenseCloud() method to findPointCloud()

- Renamed Chunk.importPoints() method to importPointCloud()
- Renamed Chunk.thinPointCloud() method to thinTiePoints()
- Renamed Chunk.triangulatePoints() method to triangulateTiePoints()
- Renamed Chunk.point_cloud attribute to tie_points
- Renamed Chunk.dense_cloud attribute to point_cloud
- Renamed Chunk.dense_clouds attribute to point_clouds
- Renamed ModelView.point_cloud_view_mode attribute to tie_points_view_mode
- Renamed ModelView.dense_cloud_view_mode attribute to point_cloud_view_mode
- Renamed AddFrames.copy_dense_cloud attribute to copy_point_cloud
- Renamed DuplicateChunk.copy_dense_clouds attribute to copy_point_clouds
- Renamed FilterPointCloud.asset attribute to point_cloud
- Renamed PublishData.save_point_colors attribute to save_point_color
- Renamed copy_dense_cloud argument in Chunk.addFrames() method to copy_point_cloud
- Renamed save_point_colors argument in Chunk.publishData() method to save_point_color
- Renamed asset argument in Chunk.filterPointCloud() method to point_cloud
- Renamed source argument in PointCloud.classifyGroundPoints() method to source_class
- Revised parameter names for point attributes in ExportPointCloud class and Chunk.exportPointCloud() methods
- Removed ImportLaserScans class
- Removed Chunk.importLaserScans() method
- Removed Chunk.samplePoints() method
- Removed use_trajectory, traj_path, traj_columns, traj_delimiter and traj_skip_rows attributes from ImportPointCloud class
- Removed use_trajectory, traj_path, traj_columns, traj_delimiter and traj_skip_rows arguments from Chunk.importPointCloud() method
- Removed merge_depth_maps, merge_dense_clouds, merge_models, merge_elevations and merge_orthomosaics attributes from MergeChunks class
- Removed merge_depth_maps, merge_dense_clouds, merge_models, merge_elevations and merge_orthomosaics arguments from Document.mergeChunks() method

3.7 Metashape version 1.8.5

- Added DetectPowerlines class
- Added Chunk.detectPowerlines() method
- Added CameraTrack.interpolate() method
- Added generic_detector, right_angle_detector, fiducials_position_corners and fiducials_position_sides attributes to DetectFiducials class
- Added archive attribute to LoadProject and SaveProject classes

- Added generic_detector, right_angle_detector, fiducials_position_corners and fiducials_position_sides arguments to Chunk.detectFiducials() method
- Added archive argument to Document.open() and Document.save() methods

3.8 Metashape version 1.8.4

- Added Shutter.Model enum
- Added ImageFormatBZ2, ImageFormatASCII and ImageFormatKTX to ImageFormat enum
- Added Shape.areaFitted() method
- Added ExportPoints.folder_depth and ExportTiledModel.folder_depth attributes
- Added ImportLaserScans.multiplane attribute
- Added folder_depth argument to Chunk.exportPoints() and Chunk.exportTiledModel() methods
- Added multiplane argument to Chunk.importLaserScans() method
- Changed type of Sensor.rolling_shutter attribute to Shutter.Model
- Fixed Antenna.location and Antenna.rotation attributes to return non-None values

3.9 Metashape version 1.8.3

- Added CloudClient class
- Added PublishData class
- Added CalibrationFormatSTMap to CalibrationFormat enum
- Reorganized arguments of Chunk.publishData() method

3.10 Metashape version 1.8.2

No Python API changes

3.11 Metashape version 1.8.1

- Added CamerasFormatMA to CamerasFormat enum
- Added global_profile attribute to ExportRaster class
- Added traj_columns, traj_delimiter, traj_path, traj_skip_rows and use_trajectory attributes to ImportPoints class
- Added global_profile argument to Chunk.exportRaster() method
- Added use_trajectory, traj_path, traj_columns, traj_delimiter and traj_skip_rows arguments to Chunk.importPoints() method
- Removed fix_pixel_aspect, fix_principal_point, and remove_distortions attributes from ConvertImages class

3.12 Metashape version 1.8.0

- Added BuildPanorama and CalculatePointNormals classes
- Added ImageFormatJXL to ImageFormat enum
- Added Cylindrical to Sensor.Type enum
- Added Chunk.buildPanorama(), Chunk.calculatePointNormals() and Chunk.filterDenseCloud() methods
- Added findCamera(), findCameraGroup(), findCameraTrack(), findDenseCloud(), findDepthMaps(), findElevation(), findMarker(), findMarkerGroup(), findModel(), findOrthomosaic(), findScalebar(), findScalebarGroup(), findSensor() and findTiledModel() methods to Chunk class
- Added NetworkClient.serverStatus() method
- Added NetworkClient.setBatchPaused() and NetworkClient.setNodePaused() methods
- Added Settings.project_absolute_paths and Settings.project_compression attributes
- Added CloseHoles.apply_to_selection attribute
- Added ConvertImages.merge_planes attribute
- Added ExportPoints.screen_space_error and ExportTiledModel.screen_space_error attributes
- Added ExportReport.font_size attribute
- Added ImportPoints.point_neighbors attribute
- Added home_point, interesting_zone, powerlines, restricted_zone and safety_zone attributes to PlanMission class
- Added apply_to_selection argument to Model.closeHoles() method
- Added file_format and max_waypoints arguments to CameraTrack.save() method
- Added screen_space_error argument to Chunk.exportPoints() and Chunk.exportTiledModel() methods
- Added font_size argument to Chunk.exportReport() method
- Added point_neighbors argument to Chunk.importPoints() method
- Removed Shape.Type enum
- Removed ExportPanorama class
- Removed has_z, type, vertex_ids and vertices attributes from Shape class
- Removed pauseBatch(), resumeBatch(), pauseNode() and resumeNode() methods from NetworkClient class
- Removed PlanMission.max_waypoints attribute
- Removed SaveProject.absolute_paths and SaveProject.compression attributes
- Removed compression and absolute_paths arguments from Document.save() method
- Changed default value of BuildTiledModel.face_count attribute to 20000
- Changed default value of face_count argument in Chunk.buildTiledModel() method to 20000

3.13 Metashape version 1.7.6

- Added Cylindrical to Sensor.Type enum

3.14 Metashape version 1.7.5

- Added ClassifyGroundPoints.erosion_radius attribute
- Added erosion_radius argument to DenseCloud.classifyGroundPoints() method

3.15 Metashape version 1.7.4

- Added ServiceCesium to ServiceType enum
- Added ImportLaserScans class
- Added Chunk.colorizeDenseCloud() and Chunk.colorizeModel() methods
- Added Chunk.exportTexture() and Chunk.importLaserScans() methods
- Added breakpoints and rates attributed to GeneratePrescriptionMap class
- Added SmoothModel.preserve_edges attribute
- Added breakpoints and rates arguments to Chunk.generatePrescriptionMap() method
- Added preserve_edges argument to Chunk.smoothModel method
- Renamed ClusteringMethod enum to ClassificationMethod
- Renamed cluster_count, clustering_method and boundary attributes in GeneratePrescriptionMap class
- Renamed cluster_count, clustering_method and boundary arguments in Chunk.generatePrescriptionMap() method
- Removed ServiceSputnik from ServiceType enum
- Removed min_value, max_value and grid_azimuth attributes from GeneratePrescriptionMap class
- Removed min_value, max_value and grid_azimuth arguments from Chunk.generatePrescriptionMap() method

3.16 Metashape version 1.7.3

- Added ModelFormatOSGT and ModelFormatLandXML to ModelFormat enum
- Added TiledModelFormatOSGT to TiledModelFormat enum
- Added CoordinateSystem.datumTransform() method
- Added DenseCloud.selectPointsByShapes() method
- Added Sensor.makeMaster() method
- Added Utils.dmat2euler() method
- Added Settings.lanuage attribute
- Added ShapeGroup.meta attribute

- Added Shapes.group attribute
- Added ExportPoints.compression attribute
- Added ExportTiledModel.model_compression attribute
- Added ImportModel.decode_udim attribute
- Added MatchPhotos.keypoint_limit_per_mpx attribute
- Added compression argument to Chunk.exportPoints() method
- Added model_compression argument to Chunk.exportTiledModel() method
- Added decode_udim argument to Chunk.importModel() method
- Added keypoint_limit_per_mpx argument to Chunk.matchPhotos() method
- Added uniform_sampling argument to Chunk.samplePoints() method

3.17 Metashape version 1.7.2

- Added ClusteringMethod enum
- Added PointsFormatSLPK to PointsFormat enum
- Added DuplicateAsset and GeneratePrescriptionMap classes
- Added Chunk.generatePrescriptionMap() method
- Added merge, operand_chunk, operand_frame and operand_asset attributes to BuildTiledModel class
- Added ExportReport.include_system_info attribute
- Added GenerateMasks.depth_threshold attribute
- Added merge, operand_chunk, operand_frame and operand_asset arguments to Chunk.buildTiledModel() method
- Added include_system_info argument to Chunk.exportReport() method
- Added depth_threshold argument to Chunk.generateMasks() method

3.18 Metashape version 1.7.1

- Removed LegacyMapping from MappingMode enum
- Removed ReduceOverlap.sensor attribute
- Removed sensor argument from Chunk.reduceOverlap() method

3.19 Metashape version 1.7.0

- Added Geometry and AttachedGeometry classes
- Added FrameStep enum
- Added ServiceType enum
- Added Chunk.importVideo(), Chunk.publishData() and Chunk.samplePoints() methods
- Added Shape.geometry and Shape.is_attached attributes
- Added alpha component to ShapeGroup.color attribute value
- Added ImportRaster.nodata_value and ImportRaster.has_nodata_value attributes
- Added MatchPhotos.filter_stationary_points attribute
- Added BuildOrthomosaic.ghosting_filter attribute
- Added attach_viewpoints, group_attached_viewpoints and horizontal_zigzags attributes to PlanMission class
- Added ReduceOverlap.sensor attribute
- Added dir argument to Application.getExistingDirectory(), getOpenFileName(), getOpenFileNames() and getSaveFileName() methods
- Added nodata_value and has_nodata_value arguments to Chunk.importRaster() method
- Added filter_stationary_points argument to Chunk.matchPhotos() method
- Added ghosting_filter argument to Chunk.buildOrthomosaic() method
- Added sensor argument to Chunk.reduceOverlap() method
- Renamed ImportMasks class to GenerateMasks
- Renamed MaskSource enum to MaskingMode
- Renamed Chunk.importMasks() method to Chunk.generateMasks()
- Removed ReduceOverlap.max_cameras attribute
- Removed max_cameras argument from Chunk.reduceOverlap() method

3.20 Metashape version 1.6.6

- Added Tasks.TransformRaster class
- Added ExportReference.precision attribute
- Added toNetworkTask() method to task classes
- Added Chunk.transformRaster() method
- Added precision argument to Chunk.exportReference() method

3.21 Metashape version 1.6.5

- Added Sensor.meta attribute

3.22 Metashape version 1.6.4

- Added Model.Vertex.confidence attribute
- Added ConvertImages.use_initial_calibration attribute
- Added image_orientation, save_invalid_matches and use_initial_calibration attributes to ExportCameras class
- Added ExportModel.save_confidence attribute
- Added crs and image_orientation attributes to ImportCameras class
- Added CalibrationFormatPhotomod to CalibrationFormat enum
- Added save_invalid_matches, use_initial_calibration and image_orientation arguments to Chunk.exportCameras() method
- Added save_confidence argument to Chunk.exportModel() method
- Added crs and image_orientation arguments to Chunk.importCameras() method
- Removed BuildUV.adaptive_resolution attribute
- Removed adaptive_resolution argument from Chunk.buildUV() method

3.23 Metashape version 1.6.3

- Added renderPreview() methods to DenseCloud, Model, Orthomosaic, PointCloud and TiledModel classes
- Added BuildUV.texture_size attribute
- Added DecimateModel.apply_to_selection attribute
- Added DetectFiducials.cameras, DetectFiducials.frames and DetectFiducials.generate_masks attributes
- Added ExportModel.embed_texture attribute
- Added clip_to_boundary attribute to ExportPoints, ExportModel, ExportTiledModel and ExportRaster classes
- Added RasterFormatGeoPackage to RasterFormat enum
- Added ShapesFormatGeoPackage to ShapesFormat enum
- Added source argument to Chunk.addSensor() method
- Added texture_size argument to Chunk.buildUV() method
- Added apply_to_selection argument to Chunk.decimateModel() method
- Added generate_masks, cameras and frames arguments to Chunk.detectFiducials() method
- Added embed_texture argument to Chunk.exportModel() method
- Added width, height, point_size and progress arguments to Chunk.renderPreview() method
- Added clip_to_boundary argument to Chunk.exportPoints(), Chunk.exportModel(), Chunk.exportTiledModel() and Chunk.exportRaster() methods

- Added meta argument to NetworkClient.createBatch() method
- Removed CalibrateLens.fit_p3 and CalibrateLens.fit_p4 attributes

3.24 Metashape version 1.6.2

- Added Application.ModelView and Application.OrthoView classes
- Added Application.removeMenuItem() method
- Added Model.transform() method
- Added PointCloud.cleanup() method
- Added Application.model_view and Application.ortho_view attributes
- Added BuildTexture.transfer_texture attribute
- Added PlanMission.min_pitch and PlanMission.max_pitch attributes
- Added columns, crs, delimiter, group_delimiters and skip_rows attributes to ImportShapes class
- Added CamerasFormatNVM to CamerasFormat enum
- Added PointsFormatPTX to PointsFormat enum
- Added ShapesFormatCSV to ShapesFormat enum
- Added transfer_texture argument to Chunk.buildTexture() method
- Added columns, crs, delimiter, group_delimiters and skip_rows arguments to Chunk.importShapes() method
- Moved ModelViewMode enum to ModelView class
- Renamed Application.console attribute to console_pane
- Renamed Application.captureModelView() method to ModelView.captureView()
- Renamed Application.captureOrthoView() method to OrthoView.captureView()
- Renamed Application.viewpoint attribute to ModelView.viewpoint
- Removed ReduceOverlap.capture_distance attribute
- Removed capture_distance argument from Chunk.reduceOverlap() method
- Changed default values of AlignCameras.reset_alignment and MatchPhotos.reset_matches attributes to False
- Changed default value of reset_alignment argument in Chunk.alignCameras() method to False
- Changed default value of reset_matches argument in Chunk.matchPhotos() method to False

3.25 Metashape version 1.6.1

- Added Application.releaseFreeMemory() method
- Added CoordinateSystem.towgs84 attribute
- Added Marker.enabled attribute
- Added BuildModel.subdivide_task attribute
- Added subdivide_task argument to Chunk.buildModel() method
- Changed default value of keep_depth argument in Chunk.buildModel() and Chunk.buildTiledModel() to True

3.26 Metashape version 1.6.0

- Added BBox, ImageCompression, RPCModel and Model.Texture classes
- Added Tasks.ImportTiledModel and Task.ColorizeModel classes
- Added CalibrationFormat and ReferencePreselectionMode enums
- Added Model.addTexture() and Model.remove() methods
- Added Model.getActiveTexture() and Model.setActiveTexture() methods
- Added NetworkClient.setMasterServer() method
- Added setClassesFilter(), setConfidenceFilter(), setSelectionFilter() and resetFilters() methods to DenseCloud class
- Added renderDepth(), renderImage(), renderMask() and renderNormalMap() methods to PointCloud, DenseCloud and TiledModel classes
- Added Chunk.renderPreview() method
- Added Utils.euler2mat() and Utils.mat2euler() methods
- Added Calibration.rpc attribute
- Added Marker.position_covariance attribute
- Added Model.textures attribute
- Added TiledModel.crs and TiledModel.transform attributes
- Added EulerAnglesPOK and EulerAnglesANK values to EulerAngles enum
- Added PointsFormatPCD to PointsFormat enum
- Added ShapesFormatGeoJSON to ShapesFormat enum
- Added RPC to Sensor.Type enum
- Added image_compression attribute to ExportOrthophotos, ExportRaster, ExportTiledModel and UndistortPhotos classes
- Added AddPhotos.load_rpc_txt attribute
- Added AlignCameras.min_image attribute
- Added BuildDenseCloud.point_confidence attribute
- Added BuildModel.vertex_confidence, BuildModel.max_workgroup_size and BuildModel.workitem_size_cameras attributes
- Added BuildTexture.source_model and BuildTexture.texture_type attributes
- Added BuildUV.adaptive_resolution attribute
- Added DecimateModel.asset attribute
- Added ExportPanorama.image_compression attribute
- Added ExportPoints.save_classes and ExportPoints.save_confidence attributes
- Added ExportTexture.texture_type attribute
- Added ExportTiledModel.crs attribute
- Added ImportCameras.image_list and ImportCameras.load_image_list attributes
- Added ImportPoints.calculate_normals attribute

- Added MatchPhotos.guided_matching and MatchPhotos.reference_preselection_mode attributes
- Added MergeChunks.merge_depth_maps, MergeChunks.merge_elevations and MergeChunks.merge_orthomosaics attributes
- Added OptimizeCameras.fit_corrections attribute
- Added TriangulatePoints.max_error and TriangulatePoints.min_image attributes
- Added endpoints argument to PointCloud.pickPoint(), DenseCloud.pickPoint(), Model.pickPoint() and TiledModel.pickPoint() methods
- Added compression argument to Image.save() method
- Added cull_faces and add_alpha arguments to Model.renderDepth() method
- Added cull_faces, add_alpha and raster_transform arguments to Model.renderImage() method
- Added cull_faces argument to Model.renderMask() method
- Added cull_faces and add_alpha arguments to Model.renderNormalMap() method
- Moved TiffCompression enum to ImageCompression class
- Renamed Tasks.UndistortPhotos class to Tasks.ConvertImages
- Renamed Chunk.estimateImageQuality() method to Chunk.analyzePhotos()
- Renamed Chunk.buildPoints() method to Chunk.triangulatePoints()
- Renamed Chunk.loadReference() method to Chunk.importReference()
- Renamed Chunk.saveReference() method to Chunk.exportReference()
- Renamed Chunk.refineModel() method to Chunk.refineMesh()
- Renamed network_distribute tasks attribute to subdivide_task
- Renamed AlignChunks.align_method attribute to method
- Renamed AlignChunks.match_downscale attribute to downscale
- Renamed AlignChunks.match_filter_mask attribute to filter_mask
- Renamed AlignChunks.match_mask_tiepoints attribute to mask_tiepoints
- Renamed AlignChunks.match_point_limit attribute to keypoint_limit
- Renamed AlignChunks.match_select_pairs attribute to generic_preselection
- Renamed BuildDenseCloud.store_depth attribute to keep_depth
- Renamed BuildModel.store_depth attribute to keep_depth
- Renamed BuildOrthomosaic.ortho_surface attribute to surface_data
- Renamed BuildTiledModel.store_depth attribute to keep_depth
- Renamed BuildUV.texture_count attribute to page_count
- Renamed CalibrateColors.data_source attribute to source_data
- Renamed CalibrateColors.calibrate_color_balance attribute to white_balance
- Renamed ClassifyGroundPoints.cls_from attribute to source_class
- Renamed ClassifyPoints.cls_from attribute to source_class
- Renamed ClassifyPoints.cls_to attribute to target_classes
- Renamed DecimateModel.target_face_count attribute to face_count

- Renamed DuplicateChunk.copy_dense_cloud attribute to copy_dense_clouds
- Renamed ClassifyPoints.copy_elevation attribute to copy_elevations
- Renamed ClassifyPoints.copy_model attribute to copy_models
- Renamed ClassifyPoints.copy_orthomosaic attribute to copy_orthomosaics
- Renamed ClassifyPoints.copy_tiled_model attribute to copy_tiled_models
- Renamed ExportCameras.bingo_export_geoin attribute to bingo_save_geoin
- Renamed ExportCameras.bingo_export_gps attribute to bingo_save_gps
- Renamed ExportCameras.bingo_export_image attribute to bingo_save_image
- Renamed ExportCameras.bingo_export_itera attribute to bingo_save_itera
- Renamed ExportCameras.bundler_export_list attribute to bundler_save_list
- Renamed ExportCameras.chan_order_rotate attribute to chan_rotation_order
- Renamed ExportCameras.coordinates attribute to crs
- Renamed ExportCameras.export_markers attribute to save_markers
- Renamed ExportCameras.export_points attribute to save_points
- Renamed ExportMarkers.coordinates attribute to crs
- Renamed ExportModel.coordinates attribute to crs
- Renamed ExportModel.export_alpha attribute to save_alpha
- Renamed ExportModel.export_cameras attribute to save_cameras
- Renamed ExportModel.export_colors attribute to save_colors
- Renamed ExportModel.export_comment attribute to save_comment
- Renamed ExportModel.export_markers attribute to save_markers
- Renamed ExportModel.export_normals attribute to save_normals
- Renamed ExportModel.export_texture attribute to save_texture
- Renamed ExportModel.export_udim attribute to save_udim
- Renamed ExportModel.export_uv attribute to save_uv
- Renamed ExportOrthophotos.write_alpha attribute to save_alpha
- Renamed ExportOrthophotos.write_kml attribute to save_kml
- Renamed ExportOrthophotos.write_world attribute to save_world
- Renamed ExportPoints.coordinates attribute to crs
- Renamed ExportPoints.data_source attribute to source_data
- Renamed ExportPoints.export_colors attribute to save_colors
- Renamed ExportPoints.export_comment attribute to save_comment
- Renamed ExportPoints.export_images attribute to save_images
- Renamed ExportPoints.export_normals attribute to save_normals
- Renamed ExportPoints.tile_height attribute to block_height
- Renamed ExportPoints.tile_width attribute to block_width

- Renamed ExportPoints.write_tiles attribute to split_in_blocks
- Renamed ExportRaster.data_source attribute to source_data
- Renamed ExportRaster.kmz_section_enable attribute to network_links
- Renamed ExportRaster.tile_width attribute to block_width
- Renamed ExportRaster.tile_height attribute to block_height
- Renamed ExportRaster.write_alpha attribute to save_alpha
- Renamed ExportRaster.write_kml attribute to save_kml
- Renamed ExportRaster.write_scheme attribute to save_scheme
- Renamed ExportRaster.write_tiles attribute to split_in_blocks
- Renamed ExportRaster.write_world attribute to save_world
- Renamed ExportRaster.xyz_level_min attribute to min_zoom_level
- Renamed ExportRaster.xyz_level_max attribute to max_zoom_level
- Renamed ExportShapes.coordinates attribute to crs
- Renamed ExportShapes.export_attributes attribute to save_attributes
- Renamed ExportShapes.export_labels attribute to save_labels
- Renamed ExportShapes.export_points attribute to save_points
- Renamed ExportShapes.export_polygons attribute to save_polygons
- Renamed ExportShapes.export_polylines attribute to save_polylines
- Renamed ExportTexture.write_alpha attribute to save_alpha
- Renamed ExportTiledModel.mesh_format attribute to model_format
- Renamed ImportMasks.method attribute to source
- Renamed ImportModel.coordinates attribute to crs
- Renamed ImportPoints.coordinates attribute to crs
- Renamed ImportReference.coordinates attribute to crs
- Renamed MatchPhotos.preselection_generic attribute to generic_preselection
- Renamed MatchPhotos.preselection_reference attribute to reference_preselection
- Renamed MatchPhotos.store_keypoints attribute to keep_keypoints
- Renamed RefineMesh.niterations attribute to iterations
- Renamed SmoothModel.apply_to_selected attribute to apply_to_selection
- Renamed TrackMarkers.frame_start attribute to first_frame
- Renamed TrackMarkers.frame_end attribute to last_frame
- Renamed processing methods arguments to match task parameters names (e.g. dx/dy -> resolution_x/resolution_y, write_xxx -> save_xxx, export_xxx -> save_xxx, import_xxx -> load_xxx, preselection_generic -> generic_preselection, preselection_reference -> reference_preselection, source_data -> data_source, etc.)
- Replaced Chunk.importDem() method with Chunk.importRaster() method
- Replaced Chunk.exportDem() and Chunk.exportOrthomosaic() methods with Chunk.exportRaster() method

- Removed Accuracy and Quality enums
- Removed Model.texture() and Model.setTexture() methods
- Removed ExportPoints.precision attribute
- Removed OptimizeCameras.fit_p3 and OptimizeCameras.fit_p4 attributes
- Removed PlanMission.max_cameras and PlanMission.use_cameras attributes
- Removed tiff_big, tiff_tiled and tiff_overviews attributes from ExportOrthophotos and ExportRaster classes
- Removed tiff_compression attribute from ExportOrthophotos, ExportRaster and UndistortPhotos classes
- Removed jpeg_quality attribute from ExportOrthophotos, ExportRaster, ExportTiledModel and UndistortPhotos classes

3.27 Metashape version 1.5.5

No Python API changes

3.28 Metashape version 1.5.4

- Added Tasks.FilterDenseCloud class
- Added TiledModel.FaceCount enum
- Added copy() method to Antenna, Calibration, ChunkTransform, CirTransform, CoordinateSystem, Document, MetaData, OrthoProjection, RasterTransform, Region, Shutter, Target, Version, Viewpoint and Vignetting classes
- Added CameraTrack.save() and CameraTrack.load() methods
- Added Chunk.reduceOverlap() method
- Added location_enabled and rotation_enabled attributes to Sensor.Reference class
- Added CameraTrack.chunk and CameraTrack.meta attributes
- Added BuildTiledModel.ghosting_filter and BuildTiledModel.transfer_texture attributes
- Added ExportPoints.network_distribute and ExportPoints.region attributes
- Added ExportTiledModel.jpeg_quality and ExportTiledModel.texture_format attributes
- Added prevent_intersections argument to Chunk.buildContours() method
- Added transfer_texture argument to Chunk.buildTiledModel() method
- Added region argument to Chunk.exportPoints() method
- Added texture_format and jpeg_quality arguments to Chunk.exportTiledModel() method
- Added progress argument to Chunk.importMarkers() method
- Added ImageFormatWebP to ImageFormat enum

3.29 Metashape version 1.5.3

- Added DepthMap.getCalibration() and DepthMap.setCalibration() methods
- Added NetworkClient.dumpBatches(), NetworkClient.loadBatches() and NetworkClient.setBatchNodeLimit() methods
- Added location_enabled and rotation_enabled attributes to Camera.Reference class
- Added keep_depth argument to Chunk.buildTiledModel() method
- Added uv argument to Chunk.exportModel() method
- Added level argument to DepthMap.image() and DepthMap.setImage() methods
- Changed default value of keep_depth argument in Chunk.buildDenseCloud() and Chunk.buildModel() methods to True
- Changed default value of max_neighbors argument in Chunk.buildDenseCloud() method to 100

3.30 Metashape version 1.5.2

- Added CameraTrack class
- Added Tasks.PlanMission and Tasks.ReduceOverlap classes
- Added Camera.Type enum
- Added Chunk.addCameraTrack() method
- Added Application.title attribute
- Added Camera.type attribute
- Added Chunk.camera_track and Chunk.camera_tracks attributes
- Added BuildModel.trimming_radius attribute
- Added DetectMarkers.filter_mask attribute
- Added ImportReference.shutter_lag attribute
- Added Bundler and BINGO specific attributes to ExportCameras class
- Added supports_gpu attribute to task classes
- Added x, y, w, h arguments to Image.open() method
- Added filter_mask argument to Chunk.detectMarkers() method
- Added image_list argument to Chunk.importCameras() method
- Added shutter_lag argument to Chunk.loadReference() method
- Added ImageFormatBIL, ImageFormatXYZ, ImageFormatDDS to ImageFormat enum
- Removed Tasks.PlanMotion class
- Removed Animation class
- Removed Chunk.animation attribute
- Removed smoothness attribute from Tasks.BuildModel and Tasks.BuildTiledModel classes
- Removed quality and reuse_depth arguments from Chunk.buildModel() method

- Removed downscale, filter_mode, max_neighbors, max_workgroup_size, network_distribute, reuse_depth, workitem_size_cameras from Tasks.BuildModel class

3.31 Metashape version 1.5.1

- Added License class
- Added Tasks.MergeAssets class
- Added Metashape.license attribute
- Renamed Tasks.OptimizeCoverage class to Tasks.PlanMotion

3.32 Metashape version 1.5.0

- Added Sensor.Reference class
- Added Tasks.ClassifyPoints and Tasks.OptimizeCoverage classes
- Added DataType enum
- Added Model.TextureType enum
- Added Tasks.TargetType enum
- Added Animation.Track.resize() method
- Added Chunk.findFrame() method
- Added DenseCloud.classifyPoints() method
- Added Document.findChunk() method
- Added Model.Faces.resize(), Model.Vertices.resize() and Model.TexVertices.resize() methods
- Added Tasks.createTask() method
- Added decode(), decodeJSON(), encodeJSON() methods to task classes
- Added Antenna.location_covariance and Antenna.rotation_covariance attributes
- Added Camera.calibration, Camera.location_covariance and Camera.rotation_covariance attributes
- Added Chunk.image_contrast attribute
- Added DenseCloud.bands and DenseCloud.data_type attributes
- Added Model.bands and Model.data_type attributes
- Added Elevation.palette attribute
- Added Model.Face.tex_index attribute
- Added Orthomosaic.bands and Orthomosaic.data_type attributes
- Added PointCloud.Point.cov attribute
- Added PointCloud.bands and PointCloud.data_type attributes
- Added Sensor.data_type, Sensor.film_camera, Sensor.location_covariance, Sensor.reference and Sensor.rotation_covariance attributes
- Added Sensor.fixed_params and Sensor.photo_params attributes

- Added TiledModel.bands and TiledModel.data_type attributes
- Added AlignChunks.markers and AlignChunks.match_mask_tiepoints attributes
- Added BuildOrthomosaic.refine_seamlines attribute
- Added DetectMarkers.cameras and DetectMarkers.maximum_residual attributes
- Added ExportModel.colors_rgb_8bit and ExportPoints.colors_rgb_8bit attributes
- Added ExportOrthophotos.tiff_tiled and ExportRaster.tiff_tiled attributes
- Added OptimizeCameras.tiepoint_covariance attribute
- Added BuildModel.smoothness and BuildTiledModel.smoothness attributes
- Added target and workitem_count attributes to task classes
- Added max_workgroup_size and workitem_size_tiles attributes to Tasks.BuildDem class
- Added max_workgroup_size and workitem_size_cameras attributes to Tasks.BuildDenseCloud class
- Added max_workgroup_size and workitem_size_cameras attributes to Tasks.BuildDepthMaps class
- Added max_workgroup_size and workitem_size_cameras attributes to Tasks.BuildModel class
- Added max_workgroup_size, workitem_size_cameras and workitem_size_tiles attributes to Tasks.BuildOrthomosaic class
- Added max_workgroup_size, workitem_size_cameras and face_count attributes attributes to Tasks.BuildTiledModel class
- Added max_workgroup_size, workitem_size_cameras and workitem_size_pairs attributes to Tasks.MatchPhotos class
- Added refine_seamlines argument to Chunk.buildOrthomosaic() method
- Added face_count argument to Chunk.buildTiledModel() method
- Added keypoints argument to Chunk.copy() method
- Added maximum_residual and cameras arguments to Chunk.detectMarkers() method
- Added tiff_tiled argument to Chunk.exportDem(), Chunk.exportOrthomosaic() and Chunk.exportOrthophotos() methods
- Added colors_rgb_8bit argument to Chunk.exportModel() and Chunk.exportPoints() methods
- Added tiepoint_covariance argument to Chunk.optimizeCameras() method
- Added confidence argument to DenseCloud.classifyPoints() method
- Added mask_tiepoints and markers arguments to Document.alignChunks() method
- Added ignore_lock argument to Document.open() method
- Added type argument to Model.setTexture() and Model.texture() methods
- Added workitem argument to Task.apply() method
- Added ModelFormatGLTF and ModelFormatX3D to ModelFormat enum
- Added Car and Manmade to PointClass enum
- Changed default value of filter argument in Chunk.buildDepthMaps() to MildFiltering
- Removed Tasks.BuildModel.visibility_mesh attribute

3.33 PhotoScan version 1.4.4

- Added AddPhotos.strip_extensions attribute
- Added ExportRaster.image_description attribute
- Added ExportShapes.export_attributes, ExportShapes.export_labels and ExportShapes.polygons_as_polylines attributes
- Added image_description argument to Chunk.exportDem() and Chunk.exportOrthomosaic() methods
- Added format, polygons_as_polylines, export_labels and export_attributes arguments to Chunk.exportShapes() method
- Added format argument to Chunk.importShapes() method
- Added RasterFormatTMS to RasterFormat enum

3.34 PhotoScan version 1.4.3

- Added Version class
- Added Tasks.DetectFiducials class
- Added Chunk.detectFiducials() method
- Added Sensor.calibrateFiducials() method
- Added CoordinateSystem.addGeoid() method
- Added PhotoScan.version attribute
- Added Sensor.normalize_to_float attribute
- Added minimum_dist attribute to Tasks.DetectMarkers class
- Added minimum_dist argument to Chunk.detectMarkers() and Utils.detectTargets() methods
- Added keypoints argument to PointCloud.copy() method
- Changed default value of adaptive_fitting argument in Chunk.alignCameras() to False

3.35 PhotoScan version 1.4.2

- Added Tasks.ColorizeDenseCloud class
- Added PointCloud.removeKeypoints() method
- Added CoordinateSystem.transformationMatrix() method
- Added Vector.cross() method
- Added Shapes.updateAltitudes() method
- Added log_enable, log_path, network_enable, network_host, network_path and network_port attributes to Application.Settings class
- Added covariance_matrix and covariance_params attributes to Calibration class
- Added flip_x, flip_y, flip_z attributes to Tasks.BuildDem and Tasks.BuildOrthomosaic classes

- Added max_neighbors attribute to Tasks.BuildDenseCloud, Tasks.BuildDepthMaps and Tasks.BuildModel classes
- Added jpeg_quality, tiff_compression and update_gps_tags attributes to Tasks.UndistortPhotos class
- Added copy_keypoints attribute to Tasks.DuplicateChunk class
- Added width, height and world_transform attributes to Tasks.ExportRaster class
- Added store_depth attribute to Tasks.BuildTiledModel class
- Added DenseCloud.crs and DenseCloud.transform attributes
- Added CoordinateSystem.wkt2 attribute
- Added keep_keypoints argument to Chunk.matchPhotos() method
- Added flip_x, flip_y, flip_z arguments to Chunk.buildDem() and Chunk.buildOrthomosaic() methods
- Added max_neighbors argument to Chunk.buildDenseCloud() and Chunk.buildDepthMaps() methods
- Added cull_faces argument to Chunk.buildOrthomosaic() method
- Added reuse_depth and ghosting_filter arguments to Chunk.buildTiledModel() method
- Added use_reflectance_panels and use_sun_sensor arguments to Chunk.calibrateReflectance() method
- Added width, height and world_transform arguments to Chunk.exportDem() and Chunk.exportOrthomosaic() methods
- Added filter_mask argument to Chunk.estimateImageQuality() method
- Added revision argument to NetworkClient.nodeList() method
- Added ImagesData to DataSource enum
- Added ModelFormatOSGB to ModelFormat enum
- Added TiledModelFormatOSGB to TiledModelFormat enum

3.36 PhotoScan version 1.4.1

- Added OrthoProjection.Type enum
- Added Camera.image() method
- Added Chunk.loadReflectancePanelCalibration() method
- Added PointCloud.Points.copy() and PointCloud.Points.resize() methods
- Added PointCloud.Projections.resize() method
- Added PointCloud.Tracks.copy() and PointCloud.Tracks.resize() methods
- Added OrthoProjection.matrix, OrthoProjection.radius and OrthoProjection.type attributes
- Added Tasks.AnalyzePhotos.filter_mask attribute
- Added Tasks.CalibrateReflectance.use_reflectance_panels and Tasks.CalibrateReflectance.use_sun_sensor attributes
- Added Tasks.MatchPhotos.mask_tiepoints attribute
- Added Tasks.OptimizeCameras.adaptive_fitting attribute
- Added strip_extensions argument to Chunk.addPhotos() method

- Added keep_depth argument to Chunk.buildDenseCloud() method
- Added adaptive_resolution argument to Chunk.buildUV() method
- Added alpha argument to Chunk.exportModel() method
- Added mask_tiepoints argument to Chunk.matchPhotos() method
- Added adaptive_fitting argument to Chunk.optimizeCameras() method
- Added mask argument to Utils.estimateImageQuality() method
- Added CamerasFormatABC and CamerasFormatFBX to CamerasFormat enum
- Added ImageFormatJP2 to ImageFormat enum
- Added LegacyMapping to MappingMode enum

3.37 PhotoScan version 1.4.0

- Added Tasks classes
- Added Animation, OrthoProjection, Target and Vignetting classes
- Added ShapesFormat enum
- Added Marker.Type enum
- Added Chunk.calibrateColors(), Chunk.calibrateReflectance() and Chunk.locateReflectancePanels() methods
- Added Chunk.buildDepthMaps(), Chunk.importPoints(), Chunk.refineModel() and Chunk.removeLighting() methods
- Added Chunk.addDenseCloud(), Chunk.addDepthMaps(), Chunk.addElevation(), Chunk.addModel(), Chunk.addOrthomosaic() and Chunk.addTiledModel() methods
- Added Chunk.sortCameras(), Chunk.sortMarkers() and Chunk.sortScalebars() methods
- Added DenseCloud.clear() method
- Added DepthMaps.clear() and DepthMaps.copy() methods
- Added Elevation.clear() and Elevation.copy() methods
- Added Model.clear() method
- Added Orthomosaic.clear() and Orthomosaic.copy() methods
- Added TiledModel.clear() and TiledModel.copy() methods
- Added Image.gaussianBlur() and Image.uniformNoise() methods
- Added NetworkTask.encode() method
- Added Utils.createChessboardImage() and Utils.detectTargets() methods
- Added Camera.Reference.location_accuracy and Camera.Reference.rotation_accuracy attributes
- Added Camera.layer_index, Camera.master and Camera.vignetting attributes
- Added Chunk.dense_clouds, Chunk.depth_maps_sets, Chunk.elevations, Chunk.models, Chunk.orthomosaics and Chunk.tiled_models attributes
- Added Chunk.animation, Chunk.camera_crs, Chunk.marker_crs and Chunk.world_crs attributes
- Added CoordinateSystem.geoccs and CoordinateSystem.geoid_height attributes

- Added Marker.Projection.valid attribute
- Added Sensor.black_level, Sensor.fiducials, Sensor.fixed_calibration, Sensor.fixed_location, Sensor.fixed_rotation, Sensor.layer_index, Sensor.location, Sensor.master, Sensor.normalize_sensitivity, Sensor.rolling_shutter, Sensor.rotation, Sensor.sensitivity and Sensor.vignetting attributes
- Added Camera.chunk, Marker.chunk, Scalebar.chunk and Sensor.chunk attributes
- Added Marker.sensor and Marker.type attributes
- Added Elevation.projection, Orthomosaic.projection and Shapes.projection attributes
- Added DenseCloud.key and DenseCloud.label attributes
- Added DepthMaps.key and DepthMaps.label attributes
- Added Elevation.key and Elevation.label attributes
- Added Model.key and Model.label attributes
- Added Orthomosaic.key and Orthomosaic.label attributes
- Added TiledModel.key and TiledModel.label attributes
- Added point_colors argument to Chunk.buildDenseCloud() method
- Added ghosting_filter argument to Chunk.buildTexture() method
- Added minimum_size argument to Chunk.detectMarkers() method
- Added raster_transform argument to Chunk.exportModel(), Chunk.exportPoints(), Chunk.exportTiledModel() methods
- Added tiff_overviews argument to Chunk.exportDem(), Chunk.exportOrthomosaic() and Chunk.exportOrthophotos() methods
- Added min_zoom_level and max_zoom_level arguments to Chunk.exportDem() and Chunk.exportOrthomosaic() methods
- Added cameras argument to Chunk.exportOrthophotos() method
- Added image_format argument to Chunk.exportPoints() method
- Added page_numbers argument to Chunk.exportReport() method
- Added items, crs, ignore_labels, threshold and progress arguments to Chunk.loadReference() method
- Added create_markers argument to Chunk.loadReference() method
- Added progress argument to Chunk.saveReference() method
- Added quality, volumetric_masks, keep_depth and reuse_depth arguments to Chunk.buildModel() method
- Added selected_faces and fix_borders arguments to Chunk.smoothModel() method
- Added export_points, export_markers, use_labels and progress arguments to Chunk.exportCameras() method
- Added channels and datatype arguments to Photo.image() method
- Added CamerasFormatBlocksExchange and CamerasFormatORIMA to CamerasFormat enum
- Added ImageFormatNone to ImageFormat enum
- Added UndefinedLayout to ImageLayout enum
- Added ModelFormatNone and ModelFormatABC to ModelFormat enum
- Added PointsFormatNone and PointsFormatCesium to PointsFormat enum
- Added RasterFormatNone to RasterFormat enum

- Added ReferenceFormatNone and ReferenceFormatAPM to ReferenceFormat enum
- Added TiledModelFormatNone, TiledModelFormatCesium and TiledModelFormatSLPK to TiledModelFormat enum
- Renamed Chunk.master_channel attribute to Chunk.primary_channel
- Removed MatchesFormat enum
- Removed Chunk.exportMatches() method
- Removed Camera.Reference.accuracy_ypr attribute
- Removed quality, filter, cameras, keep_depth, reuse_depth arguments from Chunk.buildDenseCloud() method
- Removed color_correction argument from Chunk.buildOrthomosaic() and Chunk.buildTexture() methods
- Removed fit_shutter argument from Chunk.optimizeCameras() method

3.38 PhotoScan version 1.3.5

No Python API changes

3.39 PhotoScan version 1.3.4

No Python API changes

3.40 PhotoScan version 1.3.3

- Added network_links argument to Chunk.exportDem() and Chunk.exportOrthomosaic() methods
- Added read_only argument to Document.open() method
- Added NetworkClient.setNodeCPUEnable() and NetworkClient.setNodeGPUMask() methods
- Added Chunk.modified, DenseCloud.modified, DepthMaps.modified, Document.modified, Elevation.modified, Masks.modified, Model.modified, Orthomosaic.modified, PointCloud.modified, Shapes.modified, Thumbnails.modified, TiledModel.modified attributes
- Added Document.read_only attribute
- Added CamerasFormatSummit to CamerasFormat enum

3.41 PhotoScan version 1.3.2

- Added vertex_colors argument to Chunk.buildModel() method
- Added Shape.vertex_ids attribute

3.42 PhotoScan version 1.3.1

- Added Settings and TiledModel classes
- Added Application.getBool() method
- Added Camera.unproject() method
- Added Chunk.addFrames(), Chunk.addMarkerGroup(), Chunk.addScalebarGroup() and Chunk.buildSeamlines() methods
- Added DenseCloud.pickPoint() and DenseCloud.updateStatistics() methods
- Added Elevation.altitude() method
- Added Matrix.svd() method
- Added Model.pickPoint() method
- Added Orthomosaic.reset() and Orthomosaic.update() methods
- Added PointCloud.pickPoint() method
- Added filter argument to Application.getOpenFileName(), Application.getOpenFileNames() and Application.getSaveFileName() methods
- Added point and visibility arguments to Chunk.addMarker() method
- Added raster_transform and write_scheme arguments to Chunk.exportDem() method
- Added write_scheme and white_background arguments to Chunk.exportOrthomosaic() method
- Added white_background argument to Chunk.exportOrthophotos() method
- Added projection argument to Chunk.exportMarkers() method
- Added markers argument to Chunk.exportModel() method
- Added pairs argument to Chunk.matchPhotos() method
- Added columns and delimiter arguments to Chunk.saveReference() method
- Added version argument to Document.save() method
- Renamed npasses argument in Chunk.smoothModel() method to strength and changed its type to float
- Renamed from and to arguments in CoordinateSystem.transform(), DenseCloud.assignClass(), DenseCloud.assignClassToSelection() and DenseCloud.classifyGroundPoints() methods to avoid collision with reserved words
- Added Application.settings attribute
- Added Chunk.tiled_model attribute
- Added ShapeGroup.color and ShapeGroup.show_labels attributes
- Added ImageFormatTGA to ImageFormat enum

3.43 PhotoScan version 1.3.0

- Added MarkerGroup, Masks, ScalebarGroup, Shutter and Thumbnails classes
- Added Application.PhotosPane class
- Added Model.Statistics class
- Added Orthomosaic.Patch and Orthomosaic.Patches classes
- Added PointCloud.Filter class
- Added CamerasFormat, EulerAngles, ImageFormat, ImageLayout, MaskOperation, MaskSource, MatchesFormat, ModelFormat, ModelViewMode, PointClass, PointsFormat, RasterFormat, ReferenceFormat, ReferenceItems, RotationOrder, TiffCompression, TiledModelFormat enums
- Added Application.captureOrthoView() method
- Added Chunk.refineMarkers() method
- Added CoordinateSystem.listBuiltinCRS() class method
- Added Matrix.translation() method
- Added Model.statistics() method
- Added NetworkClient.serverInfo(), NetworkClient.nodeStatus(), NetworkClient.setNodeCapability() and NetworkClient.quitNode() methods
- Added Photo.imageMeta() method
- Added Shape.area(), Shape.perimeter2D(), Shape.perimeter3D() and Shape.volume() methods
- Added Utils.createMarkers() method
- Added source argument to Application.captureModelView() method
- Added image_format argument to Chunk.exportDem() mehod
- Added write_alpha argument to Chunk.exportOrthophotos() method
- Added image_format and write_alpha arguments to Chunk.exportOrthomosaic() method
- Added groups, projection, shift and progress arguments to Chunk.exportShapes() method
- Added items and progress arguments to Chunk.copy() method
- Added sensor argument to Chunk.addCamera() method
- Added layout argument to Chunk.addPhotos() method
- Added jpeg_quality argument to Chunk.exportOrthomosaic() and Chunk.exportOrthophotos() methods
- Added fill_holes argument to Chunk.buildOrthomosaic() method
- Added fit_shutter argument to Chunk.optimizeCameras() method
- Added settings argument to Chunk.exportReport() method
- Added progress argument to various DenseCloud methods
- Added from argument to DenseCloud.classifyGroundPoints() method
- Added chunks and progress arguments to Document.append() method
- Added progress argument to Document.alignChunks() and Document.mergeChunks() methods
- Added revision argument to NetworkClient.batchList(), NetworkClient.batchStatus() methods

- Added Application.photos_pane attribute
- Added Camera.shutter attribute
- Added Chunk.masks and Chunk.thumbnails attributes
- Added Chunk.marker_groups and Chunk.scalebar_groups attributes
- Added Chunk.euler_angles and Chunk.scalebar_accuracy attributes
- Added CoordinateSystem.name attribute
- Added Marker.group and Scalebar.group attributes
- Added Orthomosaic.patches attribute
- Added RasterTransform.false_color attribute
- Added Sensor.bands attribute
- Added Shape.attributes attribute
- Added DepthMapsData, TiledModelData and OrthomosaicData to DataSource enum
- Added CircularTarget14bit to TargetType enum
- Renamed CameraReference class to Camera.Reference
- Renamed ConsolePane class to Application.ConsolePane
- Renamed MarkerProjection class to Marker.Projection
- Renamed MarkerProjections class to Marker.Projections
- Renamed MarkerReference class Marker.Reference
- Renamed MeshFace class to Model.Face
- Renamed MeshFaces class to Model.Faces
- Renamed MeshTexVertex class to Model.TexVertex
- Renamed MeshTexVertices class to Model.TexVertices
- Renamed MeshVertex class to Model.Vertex
- Renamed MeshVertices class to Model.Vertices
- Renamed PointCloudCameras class to PointCloud.Cameras
- Renamed PointCloudPoint class to PointCloud.Point
- Renamed PointCloudPoints class to PointCloud.Points
- Renamed PointCloudProjection class to PointCloud.Projection
- Renamed PointCloudProjections class to PointCloud.Projections
- Renamed PointCloudTrack class to PointCloud.Track
- Renamed PointCloudTracks class to PointCloud.Tracks
- Renamed ScalebarReference class to Scalebar.Reference
- Renamed ShapeVertices class to Shape.Vertices
- Renamed Application.enumOpenCLDevices() method to Application.enumGPUDevices()
- Renamed Shape.boundary attribute to Shape.boundary_type
- Renamed Chunk.accuracy_cameras to Chunk.camera_location_accuracy

- Renamed Chunk.accuracy_cameras_ypr to Chunk.camera_rotation_accuracy
- Renamed Chunk.accuracy_markers to Chunk.marker_location_accuracy
- Renamed Chunk.accuracy_projections to Chunk.marker_projection_accuracy
- Renamed Chunk.accuracy_tiepoints to Chunk.tiepoint_accuracy
- Renamed method argument in Chunk.importMasks() method to source and changed its type to MaskSource
- Replaced preselection argument with generic_preselection and reference_preselection arguments in Chunk.matchPhotos() method
- Replaced fit_cxcy argument with fit_cx and fit_cy arguments in Chunk.optimizeCameras() method
- Replaced fit_k1k2k3 argument with fit_k1, fit_k2 and fit_k3 arguments in Chunk.optimizeCameras() method
- Replaced fit_p1p2 argument with fit_p1 and fit_p2 arguments in Chunk.optimizeCameras() method
- Replaced Application.cpu_cores_inactive with Application.cpu_enable attribute
- Changed type of source_data argument in Chunk.buildContours() to DataSource
- Changed type of format argument in Chunk.importCameras() and Chunk.exportCameras() methods to CamerasFormat
- Changed type of rotation_order argument in Chunk.exportCameras() to RotationOrder
- Changed type of format argument in Chunk.exportDem() and Chunk.exportOrthomosaic() methods to RasterFormat
- Changed type of format argument in Chunk.exportMatches() method to MatchesFormat
- Changed type of texture_format argument in Chunk.exportModel() method to ImageFormat
- Changed type of format argument in Chunk.importModel() and Chunk.exportModel() methods to ModelFormat
- Changed type of format argument in Chunk.exportPoints() method to PointsFormat
- Changed type of tiff_compression argument in Chunk.exportOrthomosaic() and Chunk.exportOrthophotos() methods to TiffCompression
- Changed type of items argument in Chunk.exportShapes() method to Shape.Type
- Changed type of format argument in Chunk.exportTiledModel() method to TiledModelFormat
- Changed type of mesh_format argument in Chunk.exportTiledModel() method to ModelFormat
- Changed type of operation argument in Chunk.importMasks() method to MaskOperation
- Changed type of format argument in Chunk.loadReference() and Chunk.saveReference() methods to ReferenceFormat
- Changed type of items argument in Chunk.saveReference() method to ReferenceItems
- Removed return values from Camera.open(), Chunk.addPhotos(), Chunk.alignCameras(), Chunk.buildContours(), Chunk.buildDem(), Chunk.buildDenseCloud(), Chunk.buildModel(), Chunk.buildOrthomosaic(), Chunk.buildPoints(), Chunk.buildTexture(), Chunk.buildTiledModel(), Chunk.buildUV(), Chunk.decimateModel(), Chunk.detectMarkers(), Chunk.estimateImageQuality(), Chunk.exportCameras(), Chunk.exportDem(), Chunk.exportMarkers(), Chunk.exportMatches(), Chunk.exportModel(), Chunk.exportOrthomosaic(), Chunk.exportOrthophotos(), Chunk.exportPoints(), Chunk.exportReport(), Chunk.exportShapes(), Chunk.exportTiledModel(), Chunk.importCameras(), Chunk.importDem(), Chunk.importMarkers(), Chunk.importMasks(), Chunk.importModel(), Chunk.importShapes(), Chunk.loadReference(), Chunk.loadReferenceExif(), Chunk.matchPhotos(), Chunk.optimizeCameras(), Chunk.remove(), Chunk.saveReference(), Chunk.smoothModel(), Chunk.thinPointCloud(), Chunk.trackMarkers(), CirTransform.calibrate(), CoordinateSystem.init(),

DenseCloud.classifyGroundPoints(), DenseCloud.compactPoints(), DenseCloud.selectMaskedPoints(), DenseCloud.selectPointsByColor(), Document.alignChunks(), Document.append(), Document.clear(), Document.mergeChunks(), Document.open(), Document.remove(), Document.save(), Mask.load(), Model.closeHoles(), Model.fixTopology(), Model.loadTexture(), Model.removeComponents(), Model.saveTexture(), Model.setTexture(), NetworkClient.abortBatch(), NetworkClient.abortNode(), NetworkClient.connect(), NetworkClient.pauseBatch(), NetworkClient.pauseNode(), NetworkClient.resumeBatch(), NetworkClient.resumeNode(), NetworkClient.setBatchPriority(), NetworkClient.setNodePriority(), Photo.open(), PointCloud.export(), RasterTransform.calibrateRange(), Thumbnail.load() methods in favor of exceptions

- Removed Chunk.exportContours() method
- Removed obsolete Matrix.diag() and Matrix.translation() class methods
- Removed unused focal_length argument from Calibration.save() method
- Modified Utils.mat2opk() and Utils.opk2mat() methods to work with camera to world rotation matrices

3.44 PhotoScan version 1.2.6

No Python API changes

3.45 PhotoScan version 1.2.5

- Added ShapeGroup and ShapeVertices classes
- Added CoordinateSystem.proj4 and CoordinateSystem.geogcs attributes
- Added Shapes.shapes and Shapes.groups attributes
- Added Shape.label, Shape.vertices, Shape.group, Shape.has_z, Shape.key and Shape.selected attributes
- Added Shapes.addGroup(), Shapes.addShape() and Shapes.remove() methods
- Added CoordinateSystem.transform() method
- Added Matrix.Diag(), Matrix.Rotation(), Matrix.Translation() and Matrix.Scale() class methods
- Added Matrix.rotation() and Matrix.scale() methods
- Added DenseCloud.restorePoints() and DenseCloud.selectPointsByColor() methods
- Added Application.captureModelView() method
- Added Mask.invert() method
- Added adaptive_fitting parameter to Chunk.alignCameras() method
- Added load_rotation and load_accuracy parameters to Chunk.loadReferenceExif() method
- Added source parameter to Chunk.buildTiledModel() method
- Added fill_holes parameter to Chunk.buildTexture() method

3.46 PhotoScan version 1.2.4

- Added NetworkClient and NetworkTask classes
- Added Calibration.f, Calibration.b1, Calibration.b2 attributes
- Added Chunk.exportMatches() method
- Added DenseCloud.compactPoints() method
- Added Orthomosaic.removeOrthophotos() method
- Added fit_b1 and fit_b2 parameters to Chunk.optimizeCameras() method
- Added tiff_big parameter to Chunk.exportOrthomosaic(), Chunk.exportDem() and Chunk.exportOrthophotos() methods
- Added classes parameter to Chunk.exportPoints() method
- Added progress parameter to processing methods
- Removed Calibration.fx, Calibration.fy, Calibration.skew attributes

3.47 PhotoScan version 1.2.3

- Added tiff_compression parameter to Chunk.exportOrthomosaic() and Chunk.exportOrthophotos() methods

3.48 PhotoScan version 1.2.2

- Added Camera.orientation attribute
- Added chunks parameter to Document.save() method

3.49 PhotoScan version 1.2.1

- Added CirTransform and RasterTransform classes
- Added Chunk.cir_transform and Chunk.raster_transform attributes
- Added Chunk.exportOrthophotos() method
- Added udim parameter to Chunk.exportModel() method
- Renamed RasterTransform enum to RasterTransformType

3.50 PhotoScan version 1.2.0

- Added Elevation and Orthomosaic classes
- Added Shape and Shapes classes
- Added Antenna class
- Added DataSource enum
- Added Camera.error() method
- Added Chunk.buildContours() and Chunk.exportContours() methods
- Added Chunk.importShapes() and Chunk.exportShapes() methods
- Added Chunk.exportMarkers() and Chunk.importMarkers() methods
- Added Chunk.importDem() method
- Added Chunk.buildDem(), Chunk.buildOrthomosaic() and Chunk.buildTiledModel() methods
- Added PointCloud.removeSelectedPoints() and PointCloud.cropSelectedPoints() methods
- Added Utils.mat2opk(), Utils.mat2ypr(), Utils.opk2mat() and Utils.ypr2mat() methods
- Added Chunk.elevation, Chunk.orthomosaic and Chunk.shapes attributes
- Added Chunk.accuracy_cameras_ypr attribute
- Added Sensor.antenna, Sensor.plane_count and Sensor.planes attributes
- Added Calibration.p3 and Calibration.p4 attributes
- Added Camera.planes attribute
- Added CameraReference.accuracy_ypr attribute
- Added CameraReference.accuracy, MarkerReference.accuracy and ScalebarReference.accuracy attributes
- Added Application.activated attribute
- Added Chunk.image_brightness attribute
- Added fit_p3 and fit_p4 parameters to Chunk.optimizeCameras() method
- Added icon parameter to Application.addMenuItem() method
- Added title and description parameters to Chunk.exportReport() method
- Added operation parameter to Chunk.importMasks() method
- Added columns, delimiter, group_delimiters, skip_rows parameters to Chunk.loadReference() method
- Added items parameter to Chunk.saveReference() method
- Renamed Chunk.exportModelTiled() to Chunk.exportTiledModel()
- Renamed Chunk.exportOrthophoto() to Chunk.exportOrthomosaic()
- Removed OrthoSurface and PointsSource enums
- Removed PointCloud.groups attribute
- Removed Chunk.camera_offset attribute

3.51 PhotoScan version 1.1.1

- Added Chunk.exportModelTiles() method
- Added noparity parameter to Chunk.detectMarkers() method
- Added blockw and blockh parameters to Chunk.exportPoints() method

3.52 PhotoScan version 1.1.0

- Added CameraOffset and ConsolePane classes
- Added CameraGroup, CameraReference, ChunkTransform, DepthMap, DepthMaps, MarkerReference, MarkerProjection, Mask, PointCloudGroups, PointCloudTrack, PointCloudTracks, ScalebarReference, Thumbnail classes
- Added Chunk.key, Sensor.key, Camera.key, Marker.key and Scalebar.key attributes
- Added Application.console attribute
- Added Application.addMenuSeparator() method
- Added Chunk.importMasks() method
- Added Chunk.addSensor(), Chunk.addCameraGroup(), Chunk.addCamera(), Chunk.addMarker(), Chunk.addScalebar() methods
- Added Chunk.addPhotos(), Chunk.addFrame() methods
- Added Chunk.master_channel and Chunk.camera_offset attributes
- Added Calibration.error() method
- Added Matrix.mulp() and Matrix.mulv() methods
- Added DenseCloud.assignClass(), DenseCloud.assignClassToSelection(), DenseCloud.removePoints() methods
- Added DenseCloud.classifyGroundPoints() and DenseCloud.selectMaskedPoints() methods
- Added Model.renderNormalMap() method
- Added DenseCloud.meta and Model.meta attributes
- Added PointCloud.tracks, PointCloud.groups attributes
- Added Image.tostring() and Image.fromstring() methods
- Added Image.channels property
- Added U16 data type support in Image class
- Added classes parameter to Chunk.buildModel() method
- Added crop_borders parameter to Chunk.exportDem() method
- Added chunk parameter to Document.addChunk() method
- Added format parameter to Calibration.save() and Calibration.load() methods
- Moved OpenCL settings into Application class
- Converted string constants to enum objects
- Removed Cameras, Chunks, DenseClouds, Frame, Frames, GroundControl, GroundControlLocations, GroundControlLocation, Markers, MarkerPositions, Models, Scalebars, Sensors classes

3.53 PhotoScan version 1.0.0

- Added DenseCloud and DenseClouds classes
- Added Chunk.exportModel() and Chunk.importModel() methods
- Added Chunk.estimateImageQuality() method
- Added Chunk.buildDenseCloud() and Chunk.smoothModel() methods
- Added Photo.thumbnail() method
- Added Image.resize() method
- Added Application.enumOpenCLDevices() method
- Added Utils.estimateImageQuality() 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
- 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
- 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
- Removed Chunk.buildDepth() method
- Removed Camera.depth() and Camera.setDepth() methods
- Removed Frame.depth() and Frame.setDepth() methods
- Removed Frame.depth_calib attribute

3.54 PhotoScan version 0.9.1

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

- Added Model.closeHoles() and Model.fixTopology() methods

3.55 PhotoScan version 0.9.0

- Added Camera, Frame and CoordinateSystem classes
- 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() and Chunk.resetDepth() methods
- Added Chunk.cameras property
- Added Utils.createDifferenceMask() method
- Revised Chunk.alignPhotos() method
- Revised Chunk.buildPoints() method
- Revised Chunk.buildModel() method
- Removed Photo class (deprecated)
- Removed GeoProjection class (deprecated)
- Removed Chunk.photos property (deprecated)

3.56 PhotoScan version 0.8.5

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

3.57 PhotoScan version 0.8.4

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

3.58 PhotoScan version 0.8.3

Initial version of PhotoScan Python API

PYTHON MODULE INDEX

m