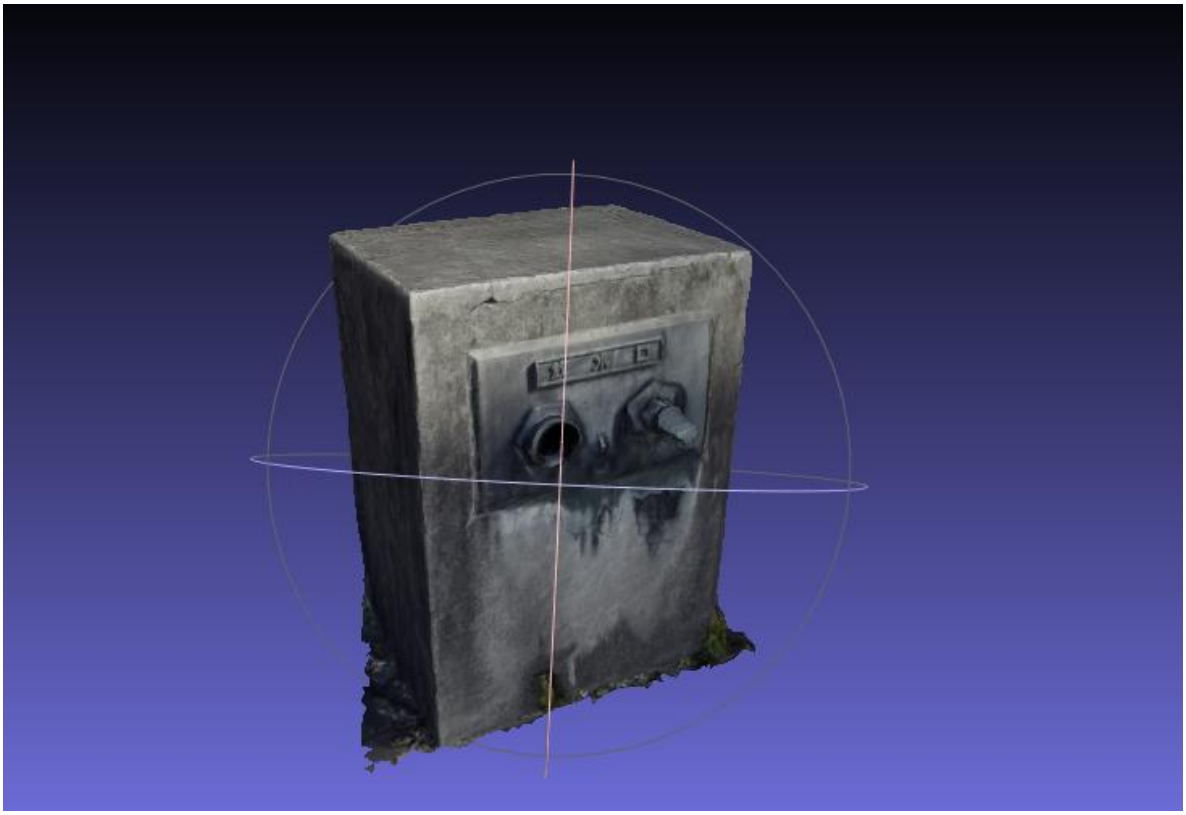
タイトル：

第 4 回 動画で 3 次元モデルつくり



「第 1 回 スマホ写真で 3 次元モデルづくり」では、画像からフォトグラメトリで 3 次元モデル生成しました。今回は、図の放水口を題材としてスマホで撮影した動画からフォトグラメトリで 3 次元モデル生成してその可視化までをご紹介します。点群データとは、無数の「点」によって構成されたデータで、メッシュデータとは、点と線で構成された無数の「面」によって構成されたデータです。

スマホで動画撮影

スマホで 3 次元モデル生成したい対象物を動画に撮影します。

画像からフォトグラメトリで 3 次元モデル生成する場合は、隣同士の画像が重なるよう (オーバーラップ) にして平行移動しながら画像を撮影するなど、画像撮影には時間や手間がかかる、もしくは撮影用アプリが必要でした。

スマホの動画から 3 次元モデル生成する場合は、特別な機種に依存しないで生成することが可能です。動画撮影する場合は、対象物の周囲を 360 度 ぐるっと回りながら周囲を撮影することで、手軽に短時間で撮影をすることが可能であり必要な角度での撮影忘れが起こりにくいメリットがあります。ただし、画像の方が動画に比べて解像度が高いことが多く高精細なデータを生成することが可能です。

スマホで動画を撮影する際のコツとしては下記があります。

- 撮影対象物は動かさない。対象物を動かすのではなく撮影者側が動くとよいです。これは対象物を手で回そうとする時の手ブレ防止と、横にずれながら撮影していく方が綺麗に 3D モデル化できるからです。
- カメラがブレないように撮影する。脇があいてしまうと安定感がなくなり手ブレしやすくなります。脇をしめることでカメラが動きにくくなり手ブレを抑えることができます。

- 急激にカメラを移動させない。一定の速度でカメラを移動させる。今回、3D モデル化のために、動画から任意の時間毎に画像を切り出します。そのため、上から下まで全ての画角に収まるように一定の速度で撮影することで、いろいろな方向と角度からの写真が取り出すことが出来ます
- カメラに対象物以外が映りこまないようにする。撮影中に背後を人が横切ったりなどすると、対象物以外の画像情報が余計な情報となり、モデル生成がうまくいかない場合があります。
- 対象物のさまざまな角度から撮影する。対象物を上から下まで全てが収まるように撮影する。表以外の上下と裏側は忘れがちなので注意しましょう。

動画から 3 次元モデルを生成

撮影した動画は、パソコンのフォトグラメトリソフトを使って 3 次元モデルを生成していき、ます。

今回は 3DF Zephyr Free (https://www.3dflow.net/3df-zephyr-free/)というフォトグラメトリソフトを使って 3 次元モデル生成する方法をご紹介します。

3DF Zephyr は有料版の 3DF Zephyr Lite、3DF Zephyr がありますが、ここでは無料版の 3DF Zephyr Free を使って 3 次元モデルを生成します。

1 ユーロ：143.71 円 2023/02/27 現在

(https://www.3dflow.net/3df-zephyr-photogrammetry-software/)

無料版の 3DF Zephyr Free の利用条件としては下記があります。

- 個人利用は無料
- 3D ビューアが利用可能
- 最大 50 枚の写真しか使用できない
- 時間の利用制限はない

3DF Zephyr は Windows 版のみ対応となります。Mac の方は iOS と Mac に対応している PhotoCatch (**https://www.photocatch.app/**)を利用すれば、同じように動画から 3 次元モデルを生成することが可能です。

※ 3DF Zephyr の有料版と無料版の比較 (**https://www.3dflow.net/3df-zephyr-feature-comparison/**)

3DF Zephyr の各ライセンス

- 3DF Zephyr Free 無料 無期限ライセンス
- 3DF Zephyr Lite (€149.00 + VAT)　21,413 円(税別)　無期限ライセンス
- 3DF Zephyr (Monthly) (€250.00 + VAT / mo)　35,928 円/月(税別) 月額ライセンス
- 3DF Zephyr (€3900.00 + VAT) 560,477 円(税別) 無期限ライセンス

3DF Zephyr Free で 3 次元モデルを生成

3DF Zephyr Free では [.mepg　.wmv　.avi　.mp4　.mov　.mpg]の動画ファイルを入力することが可能です。

ただし、撮影した動画形式によって入力できない動画があります。同じ拡張子でも、動画の映像や音声等の構成要素が異なる場合があります。入力できない場合は VLC メディアプレーヤー

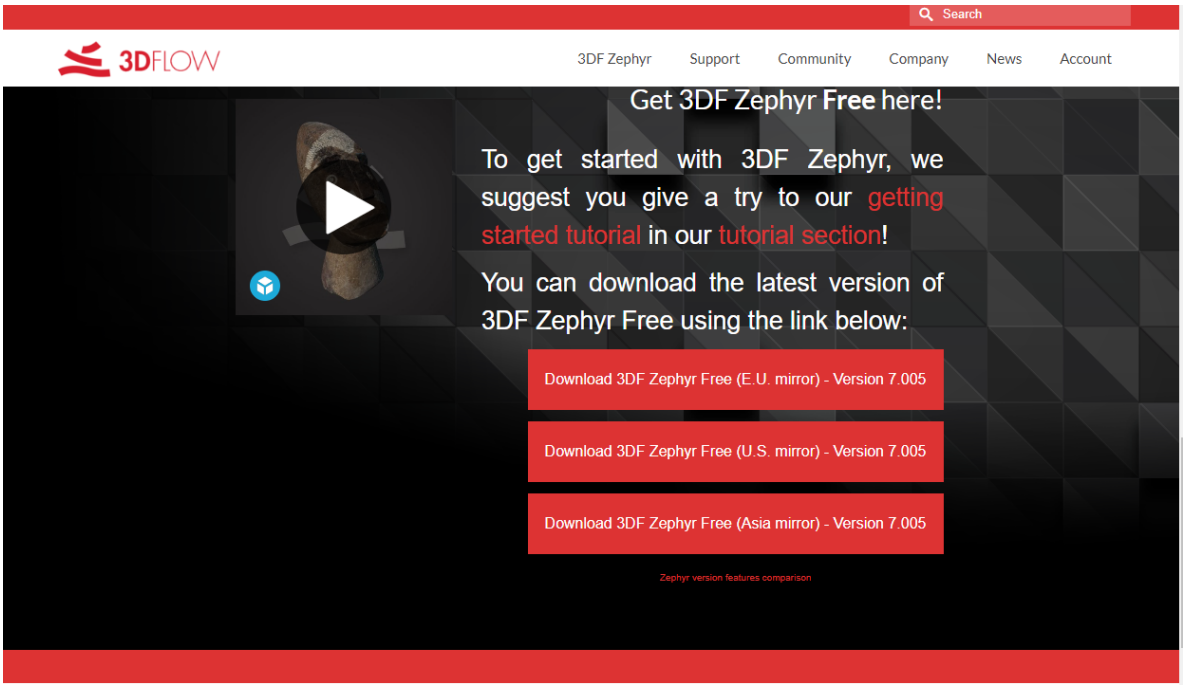
(https://www.videolan.org/vlc/index.ja.html)などで、MP4（Video H.264 + MP3(MP4)）形式の動画に変換してください。

MP4 形式の動画への変換方法は下記のサイトなどが参考になります。

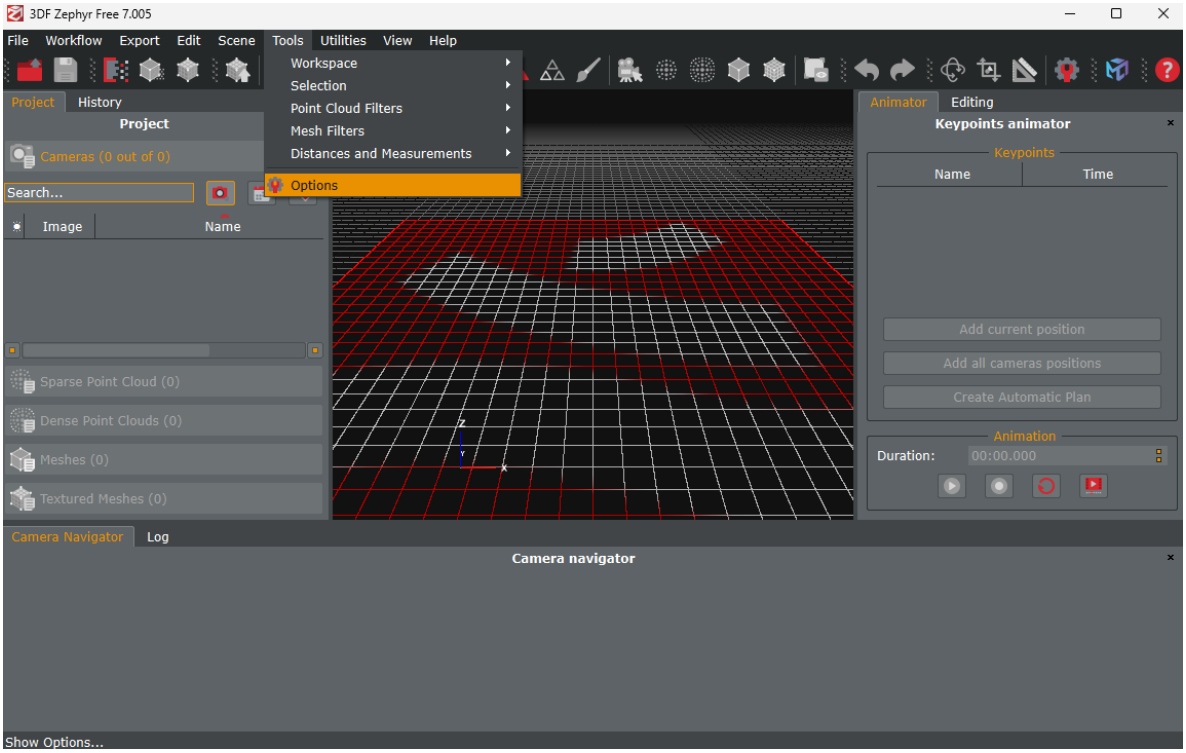
- VLC メディアプレーヤーで「MOV ファイル」を「MP4 ファイル」に変換して、Windows10 で再生できるようにする（http://customize.komaxy.com/article/182165925.html）

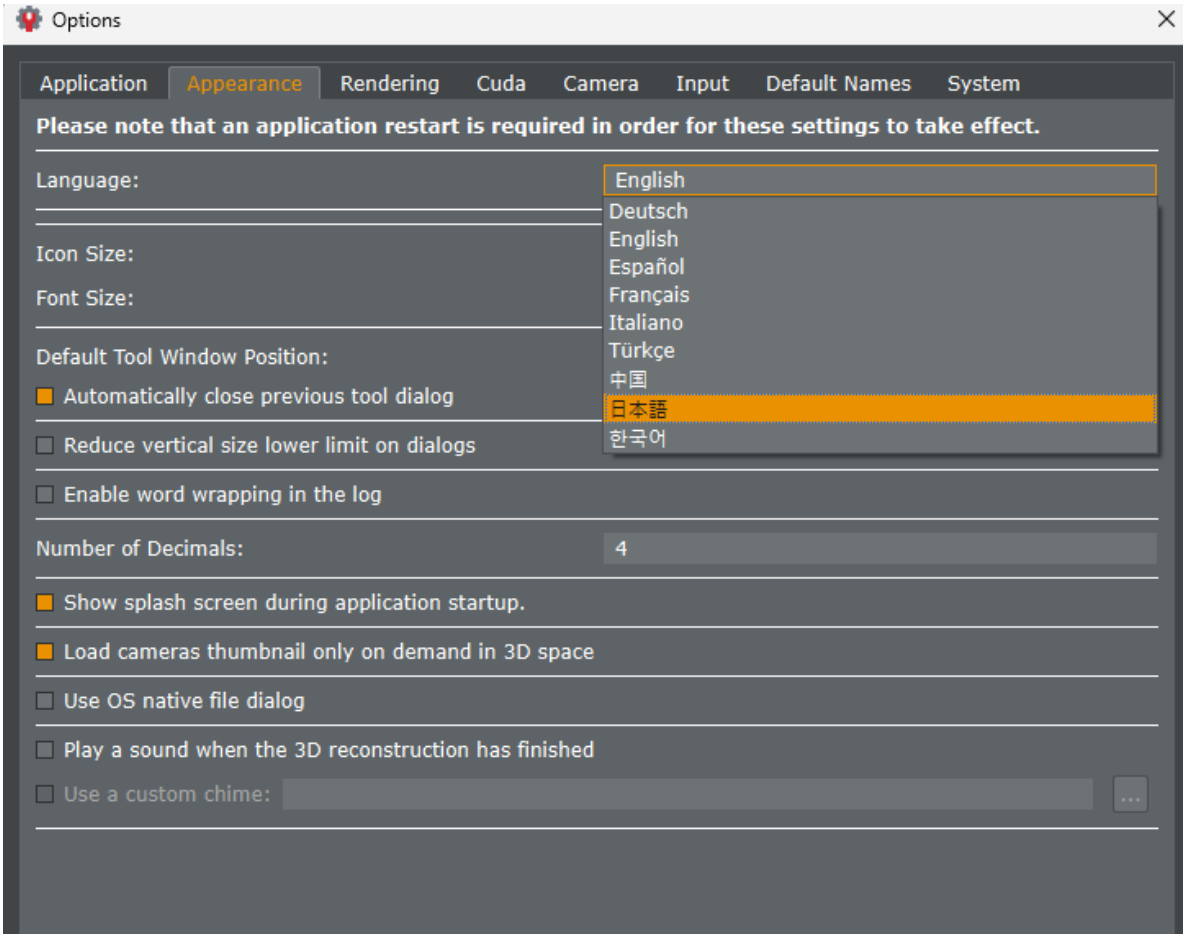
3DF Zephyr のインストールと設定

3DF Zephyr Free のインストールと設定をしていきます。



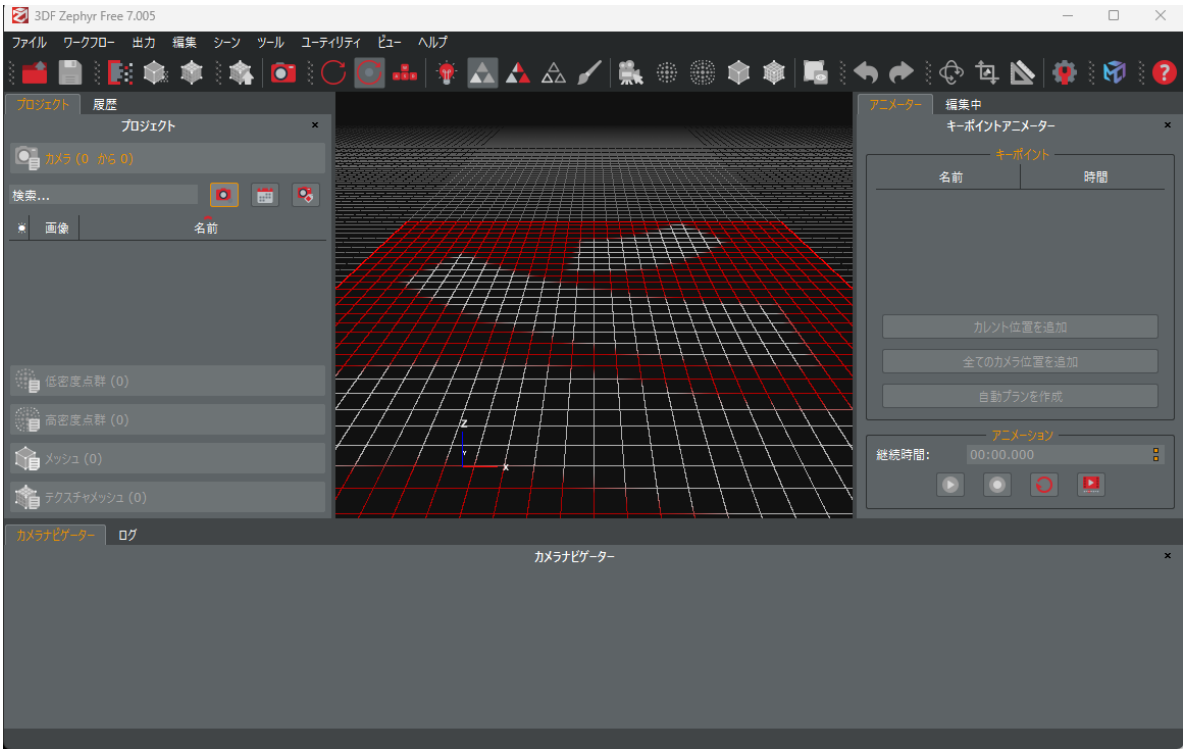
3DF Zephyr Free Ver7.005 (https://www.3dflow.net/3df-zephyr-free/)から Asia のリンクからダウンロードしてアプリをインストールします。


3DF Zephyr Free 7.005
File
Workflow
Export
Edit
Scene
Tools
Utilities
View
Help
Workspace
Selection
Point Cloud Filters
Mesh Filters
Distances and Measurements
Options
Project
History
Project
Cameras (0 out of 0)
Search...
Image
Name
Sparse Point Cloud (0)
Dense Point Clouds (0)
Meshes (0)
Textured Meshes (0)
Animator
Editing
Keypoints animator
Keypoints
Name
Time
Add current position
Add all cameras positions
Create Automatic Plan
Animation
Duration:
00:00.000
Camera Navigator
Log
Camera navigator
Show Options...

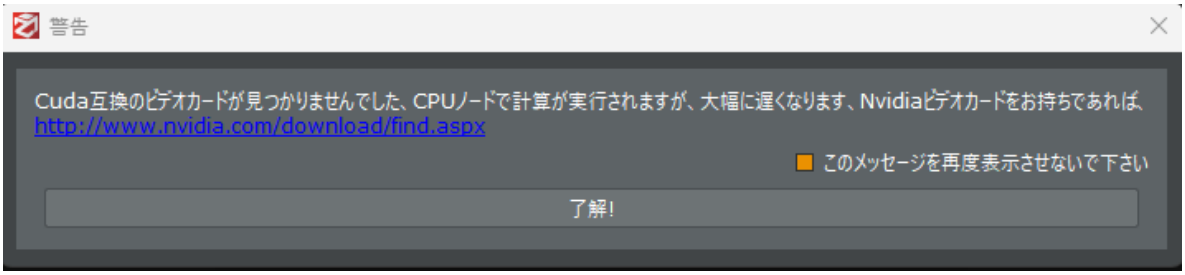


アプリが英語表記になっていたら、日本語表示に変更する為に、メニューから「Tools」→

「Options」→ 「Appearance」→ 「Language」→ 「日本語」を選択してアプリを再起動します。



アプリを再起動すると日本語表記に変更されます。



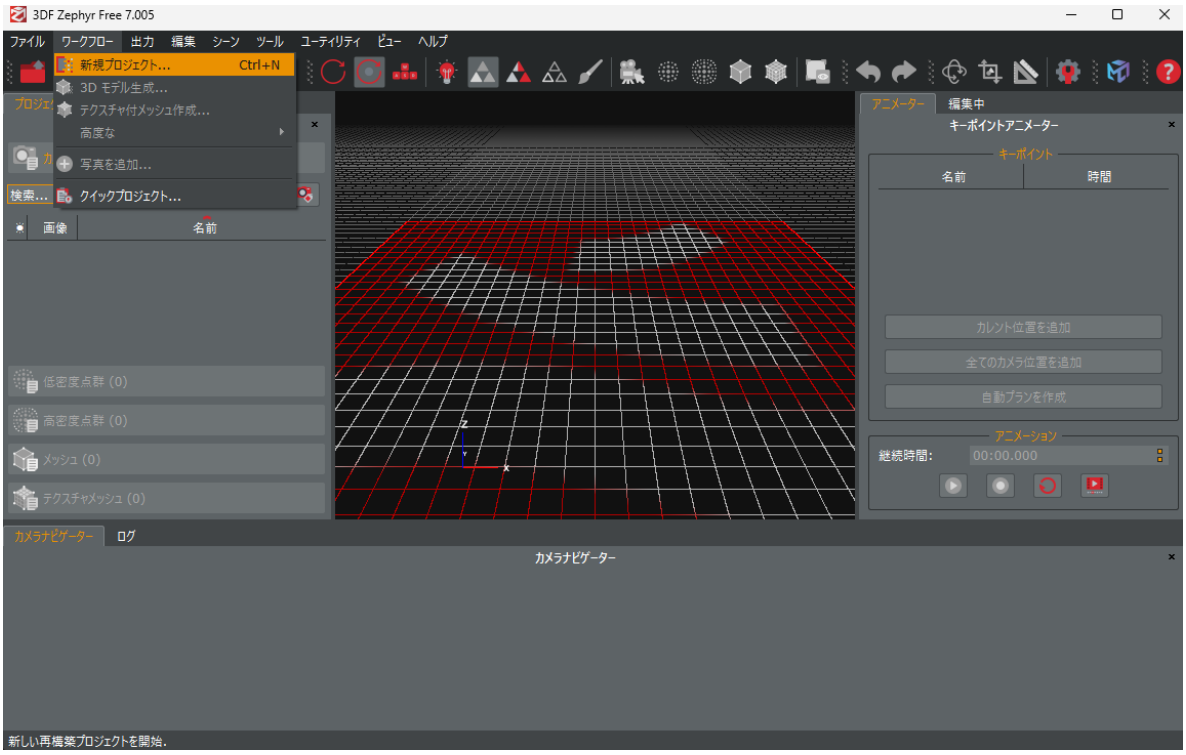
画像処理を高速に処理する GPU ハードウェアがみつからない場合は、ソフトウェアで画像処理をするため終了までの処理時間がかかります。

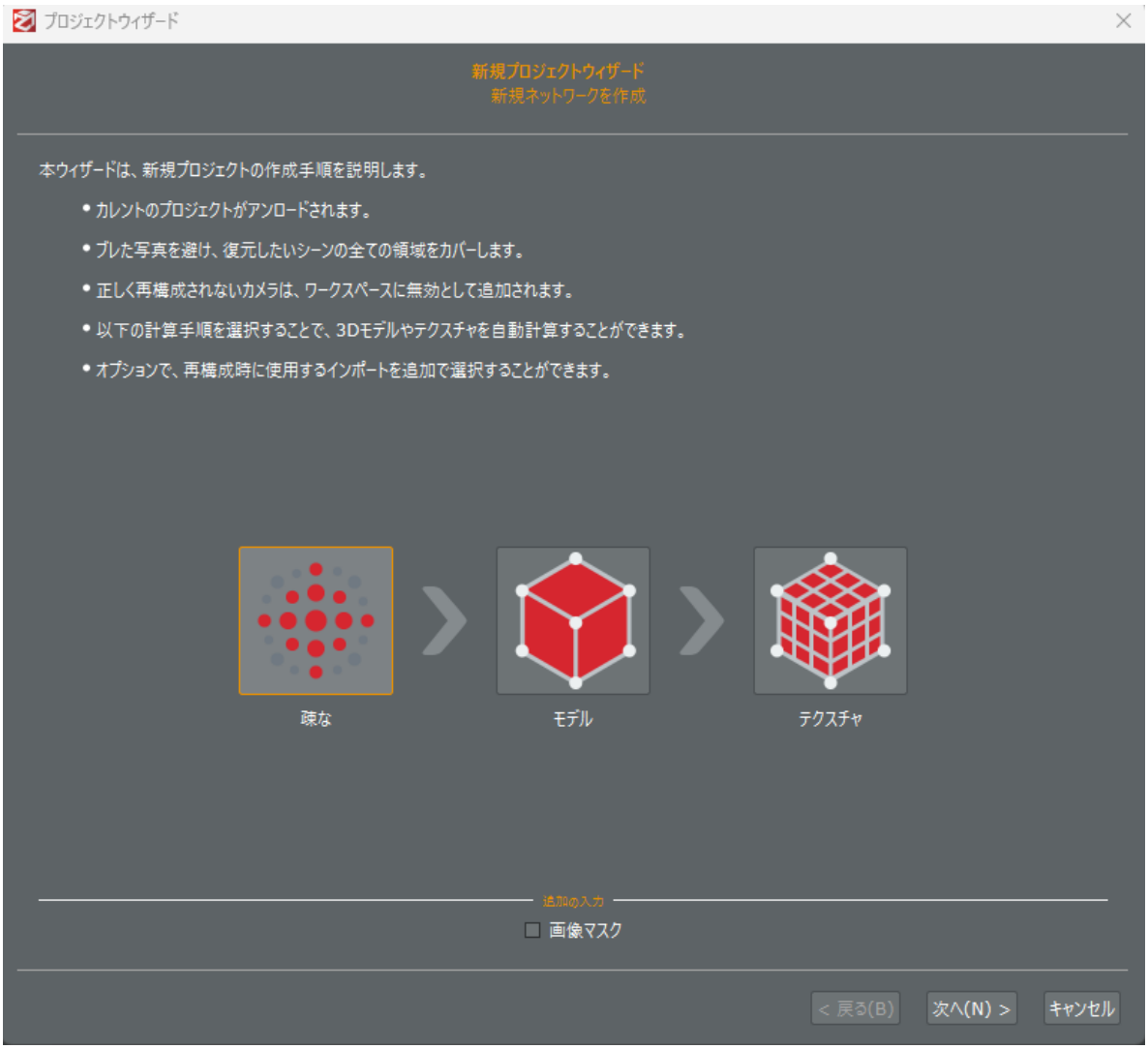
GPU ハードウェアが無いパソコンで、このメッセージが表示されても動作には問題はありません。

3DF Zephyr へ動画ファイルの取り込み

動画ファイルを 3DF Zephyr へ取り込みます。

スマホで撮影した動画ファイルをパソコンにコピーして、動画から 3 次元モデルを生成する為に動画から画像を切り出します。

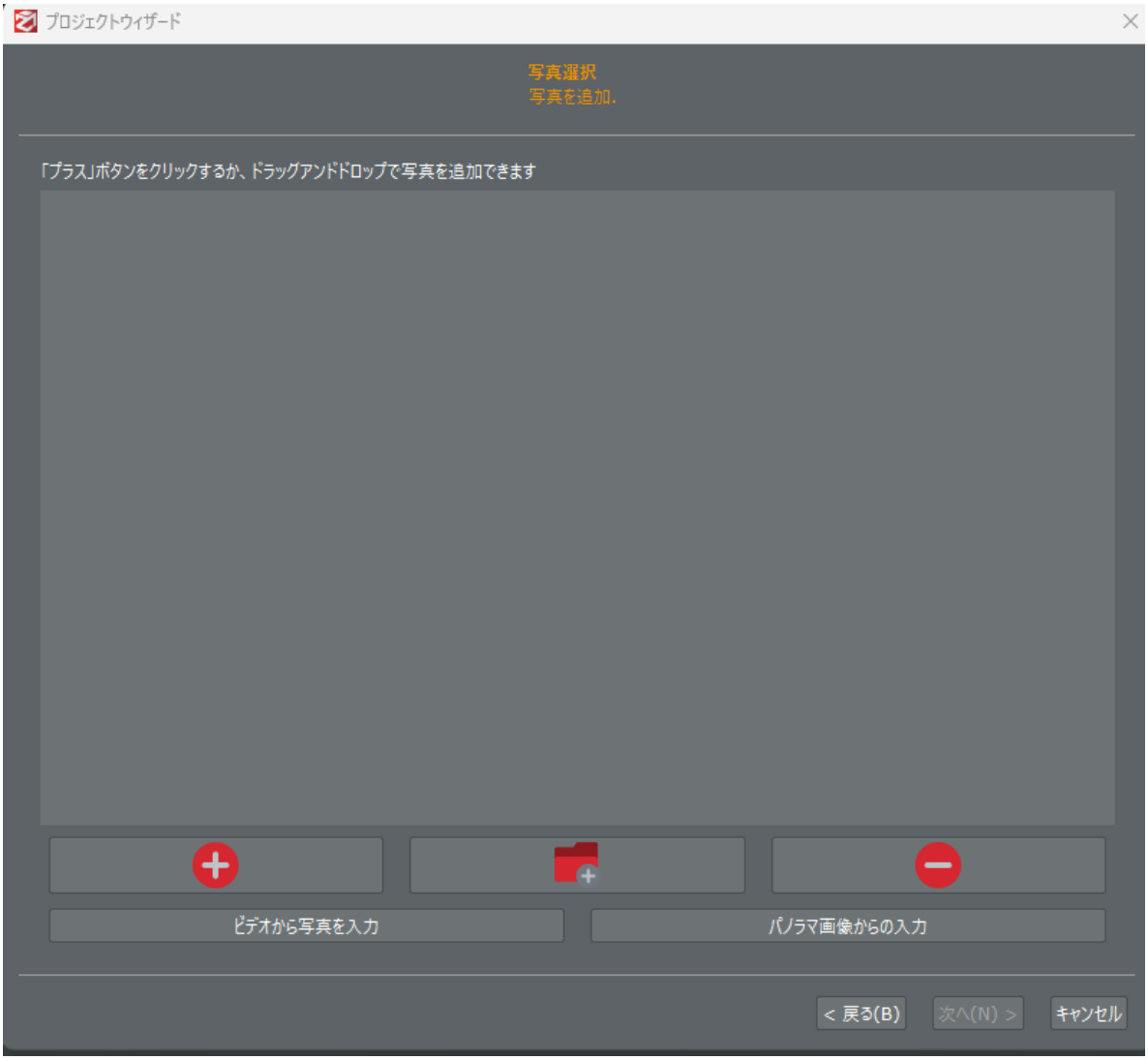




はじめに、アプリのメニューの「ワークフロー」から「新規プロジェクト」を選択すると、「プロジェクトウィザード」の「新規プロジェクトウィザード」が表示されるので「画像マスク」は OFF で「次へ」をクリックします。

「画像マスク」は、入力画像から一部分を切り抜くマスク処理（マスキング）をする場合にチェックします。マスク処理するには、切り抜きたい部分の輪郭画像（マスク画像）を別途用意し入力する必要があります。今回は動画から抽出した画像をそのまま使用す

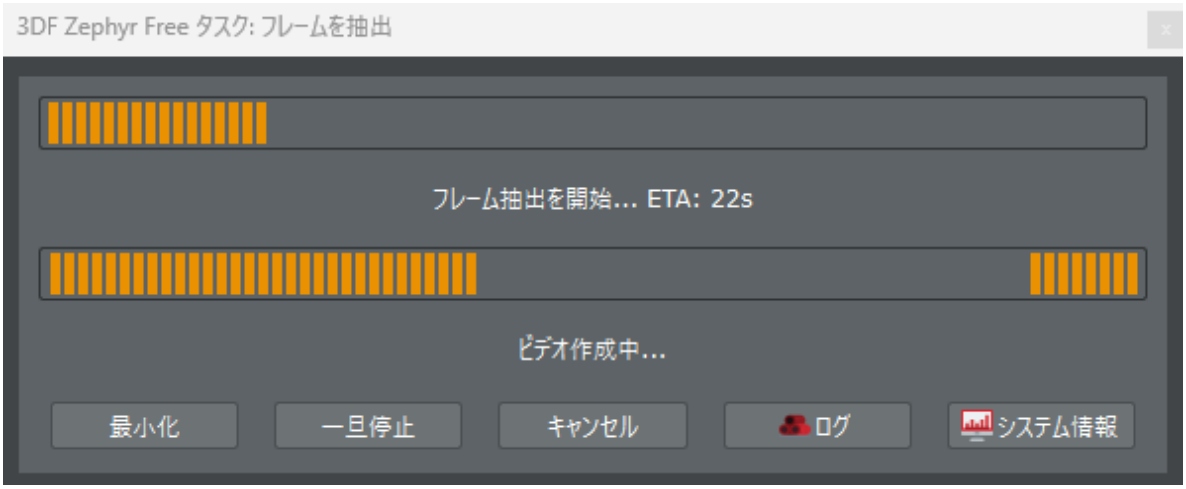
るので「画像マスク」は OFF にします。輪郭画像（マスク画像）には白黒画像を用いるのが一般的です。



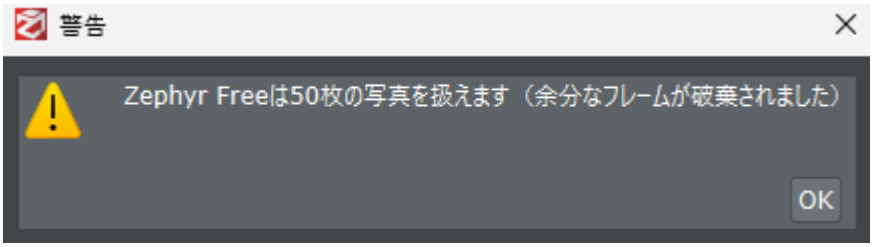
「写真選択」ウィンドウが表示されてたら「ビデオから写真を入力」を選択すると「ビデオファイルからフレームを抽出」ウィンドウが表示されるので、「入力ファイル」にスマホで撮影した動画ファイル、「出力ファイル」に動画から切り出した画像ファイルの保存先を指定します。

サンプルの動画ファイルは、38 秒（52.9 MB）の動画となります

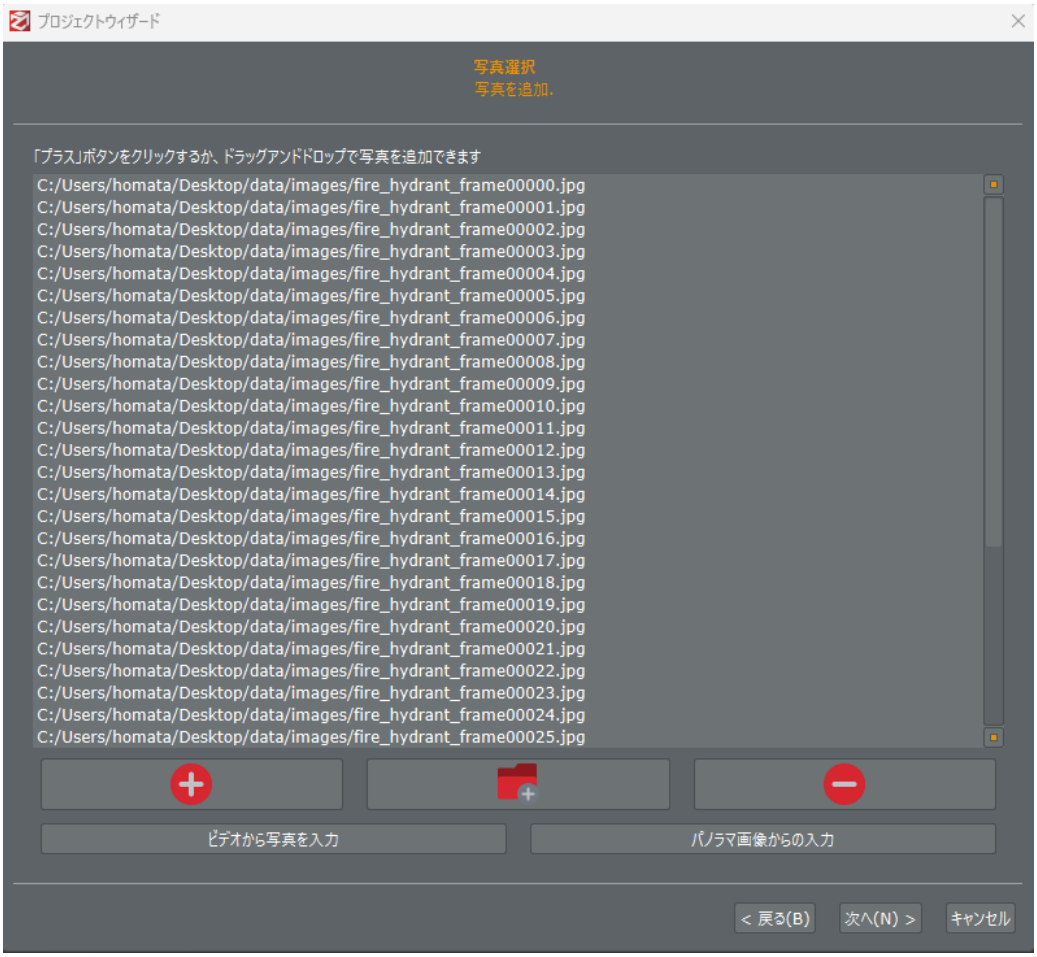
「フレームを抽出」はサンプルレートを選択して「fps (Frame per second)」を設定します。fps は 動画の 1 秒あたりの静止画枚切り出す枚数を意味しています。例えば 1.00fps に設定すると 1 秒間隔で動画から画像を切り出します。Free 版では 50 枚の制限があるので、全体の配分を fps で調整してください。fps の間隔が短いと動画の前半部分だけで終了してしまいます



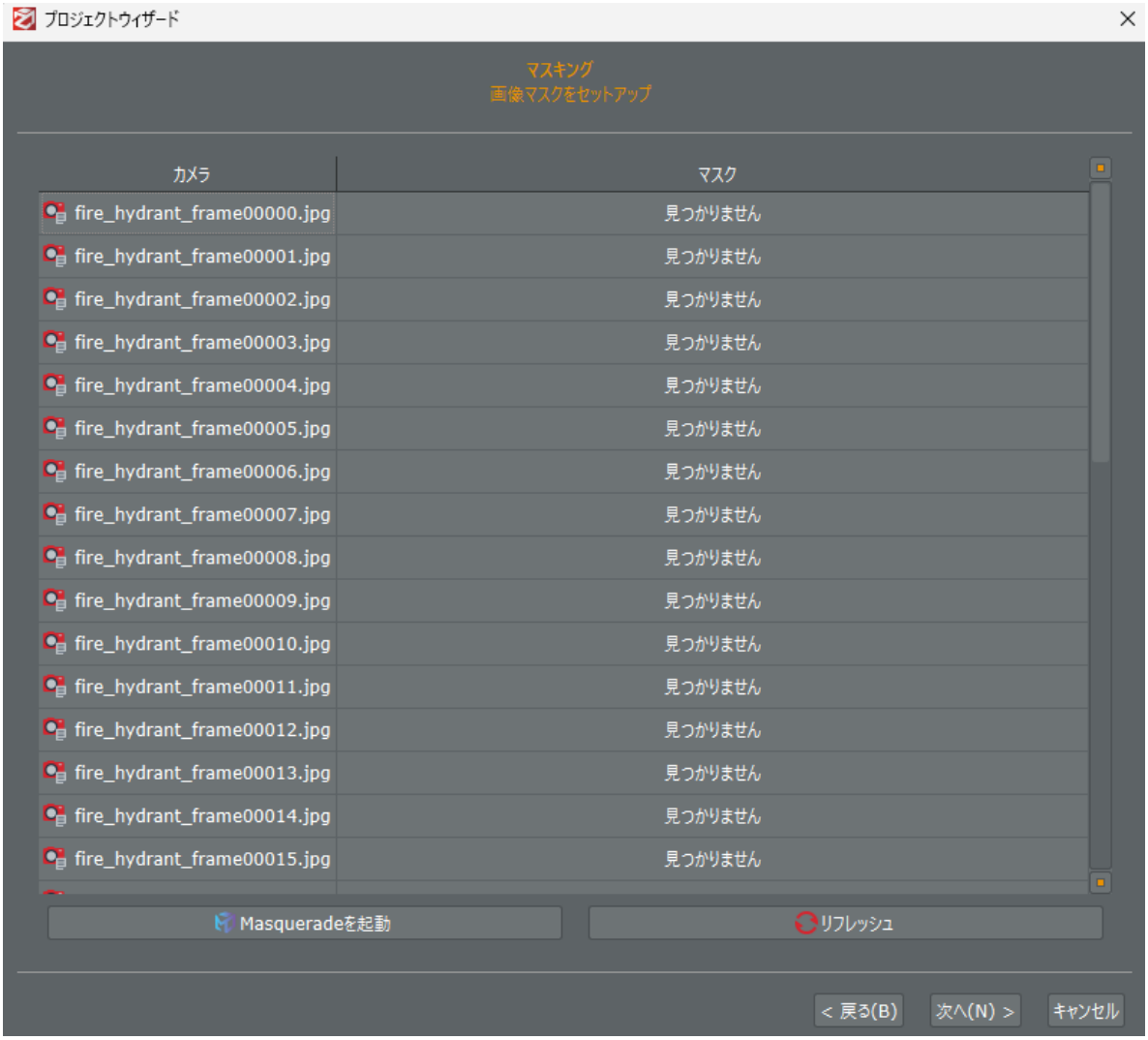
「フレームの抽出とワークスペースの入力」ボタンとクリックすると動画から画像の切り出しが開始されます。



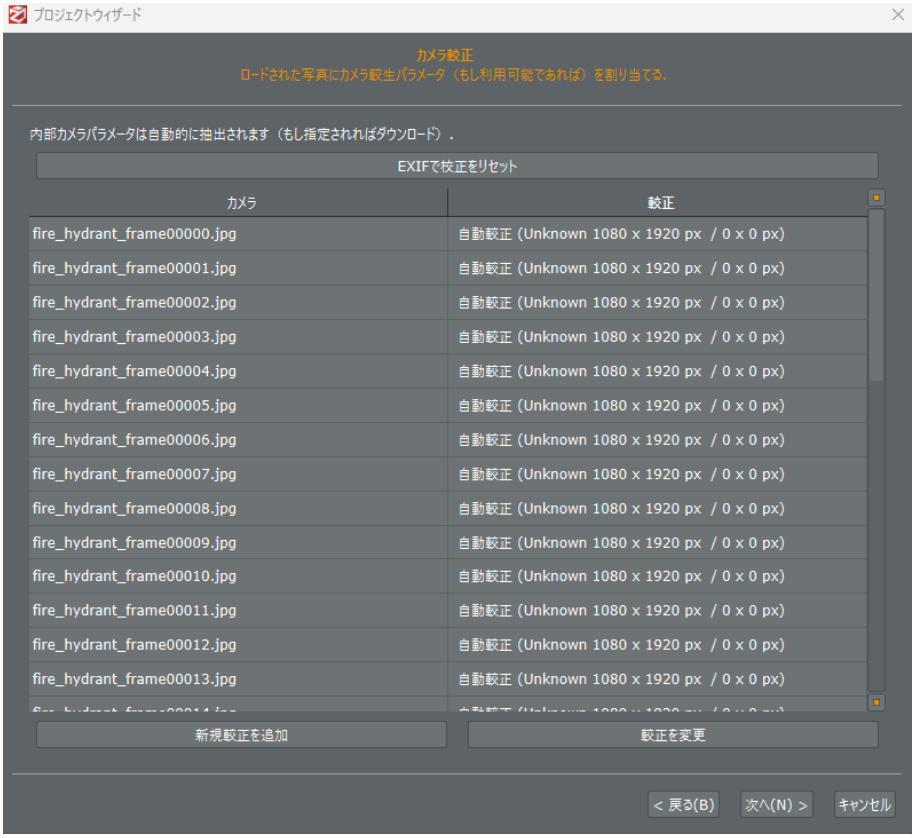
3DF Zephyr Free 版は、最大 50 フレームまでの制限があるので、画像の切り出しも 50 枚までしか出来ません。サンプルレートの fps を調整して切り出す画像を 50 枚以下になるように調整をします。切り出す画像が 50 枚以上になると処理は自動で停止するのでキャンセルして画像を 50 枚以下になるように再調整してください。入力する動画が長く fps を大きくしないと 50 枚を超える場合は、動画の長さを再調整することも必要です



動画から切り出した画像の準備ができたら「次へ」をクリックします

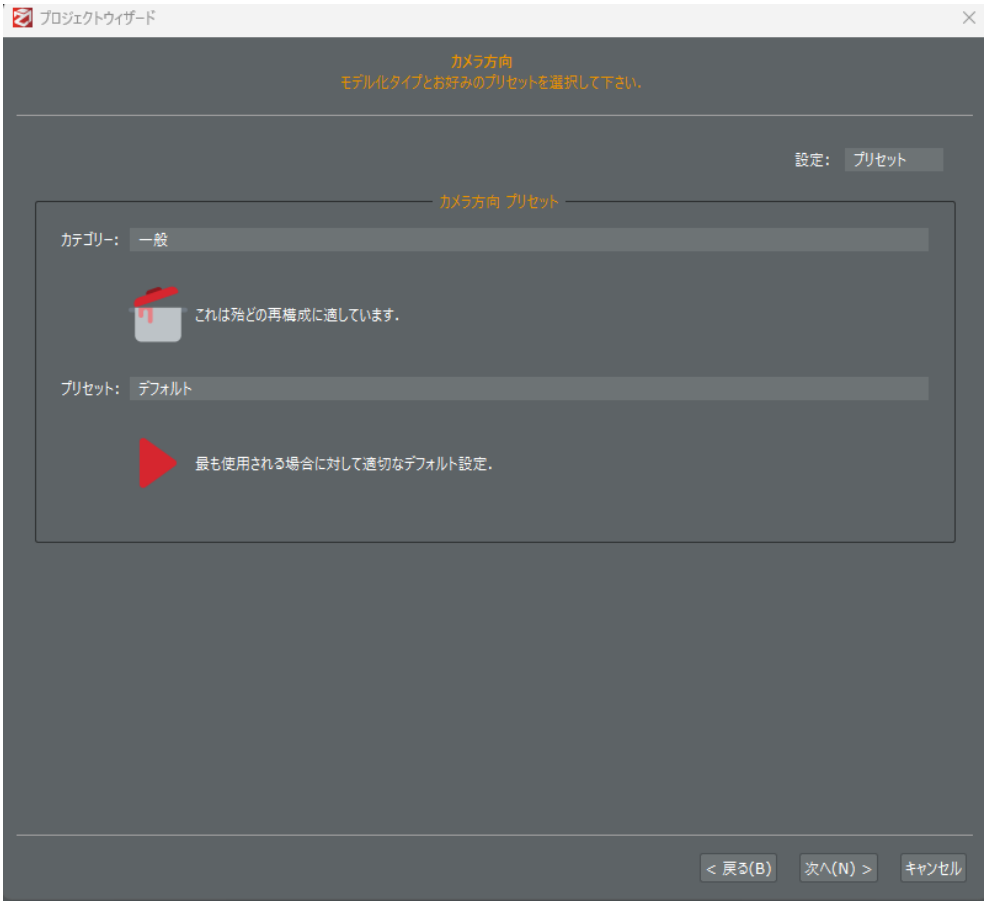


最初の「新規プロジェクトウィザード」の画面で「画像マスク」を ON にした場合、「画像マスク」の画像を入力するダイアログが表示されますが、動画から抽出した画像をそのまま使用する場合はマスク画像を指定しないで「次へ」をクリックしてください。



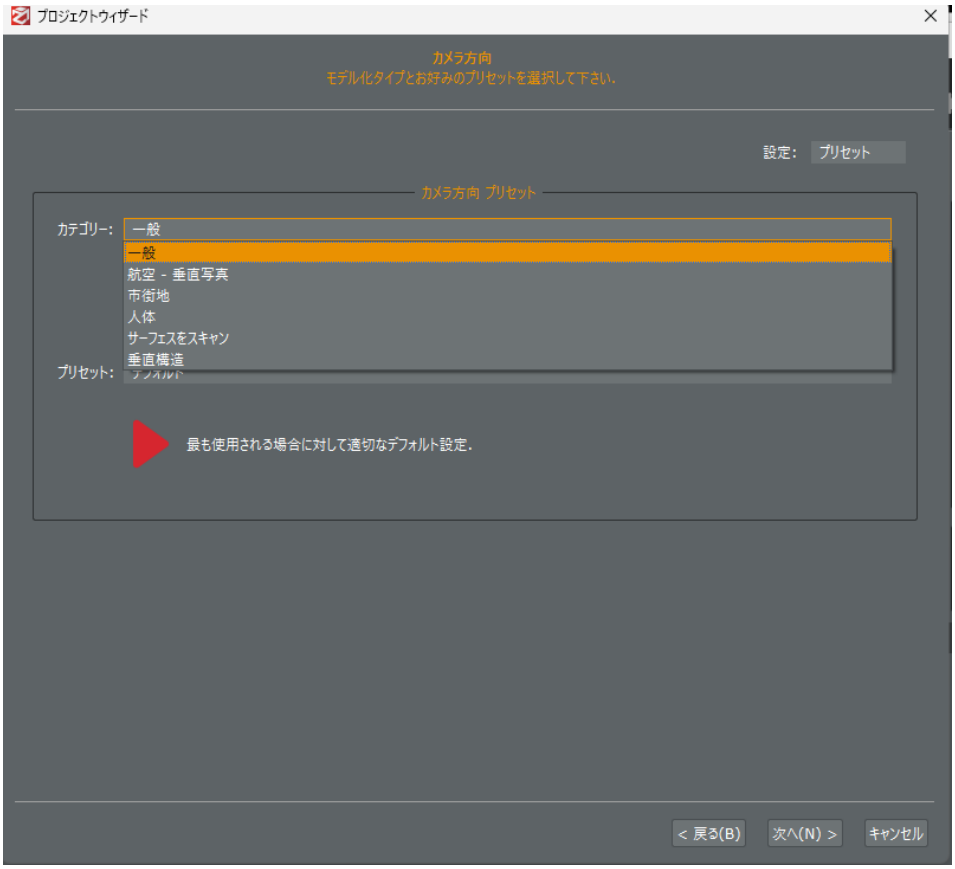
較正（こうせい、calibration）ウィンドウが表示されますが、これはカメラで撮影した画像の光の屈折補正をする画面ですが、今回は自動較正のままで「次へ」をクリックします。

カメラ較正は画像から自動で設定されます。撮影に使用したカメラの特性にあわせてより細かく設定を変更することができます。設定には専門的な知識が必要なので今回はデフォルト設定を使用します。



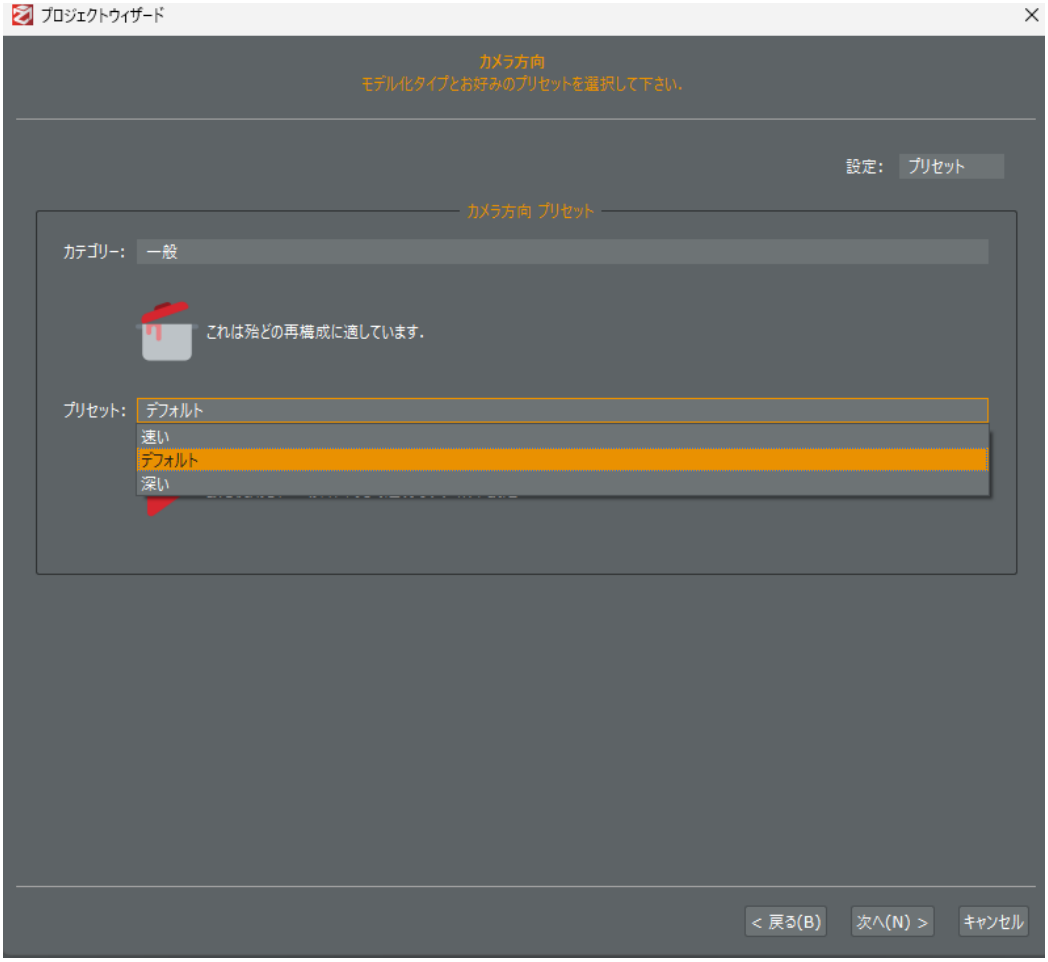
「カメラの方向」では、モデル化する被写体の種類を設定します。通常はデフォルトの「カテゴリー：一般」「プリセット：デフォルト」を選択します。

カテゴリーで「対象物の大きさや状況」、プリセットで「精度」を選択します。慣れないうちはそれぞれの項目について、カテゴリーは一般、プリセットはデフォルトに設定してください。



被写体によってカテゴリーを変更して選択します。

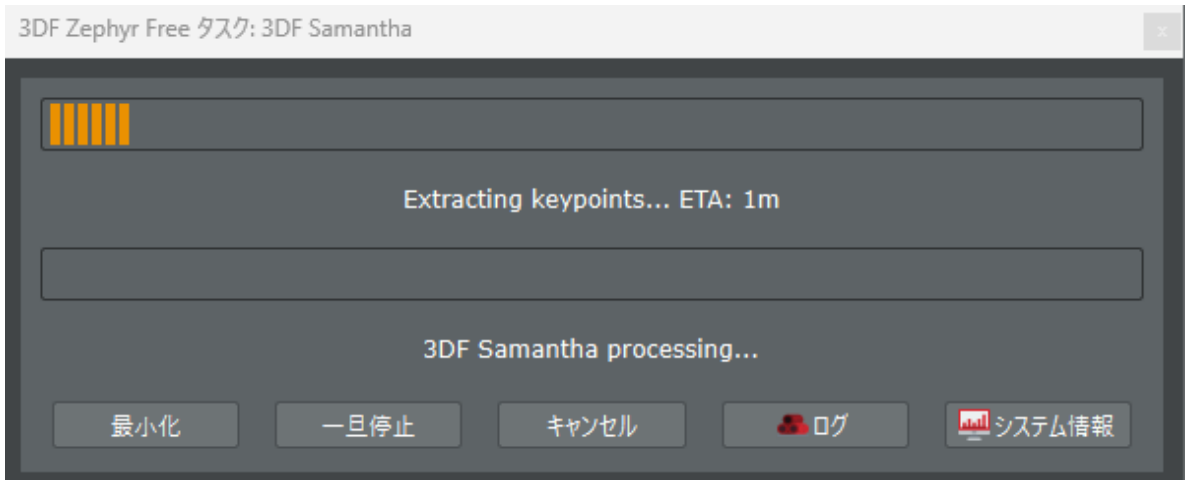
- 一般：一般的な場合に選択します
- 航空 - 垂直写真：ドローンや航空写真の場合に選択します
- 市街地：街の様々は被写体が混在している場合に選択します
- 人体：人が被写体の場合に選択します
- サーフェスをスキャン：近い位置から撮影した場合に選択します
- 垂直構造：通信塔やドローンで真上から撮った場合に選択します



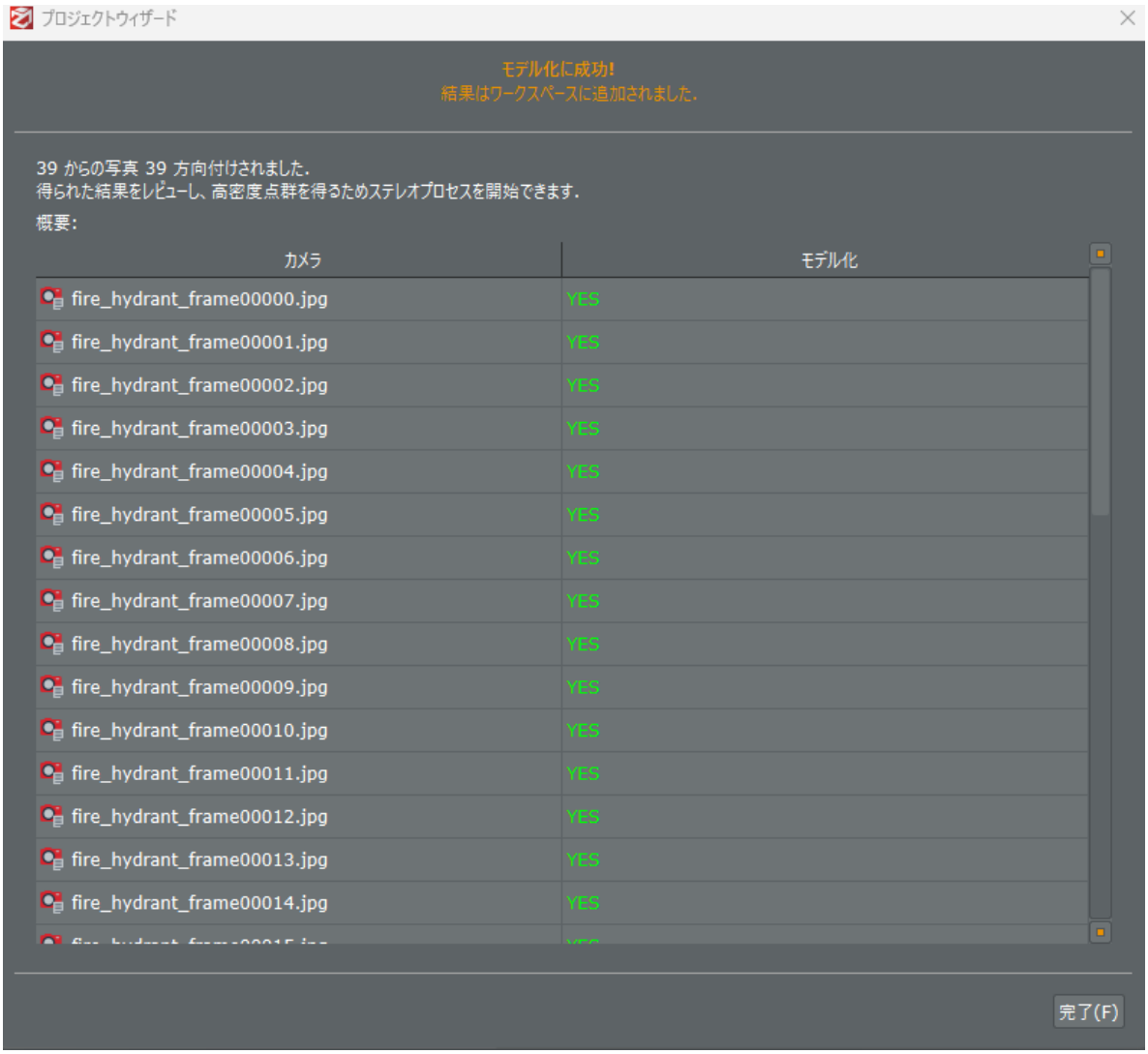
プリセットを選択します

- 速い：解像度を削減し、デフォルト設定より処理が高速なプリセット
- デフォルト：殆どの場合（ユースケース）に適したプリセット
- 深い：画像のマッチングの調整を詳細にします。デフォルト設定より処理が低速なプリセット

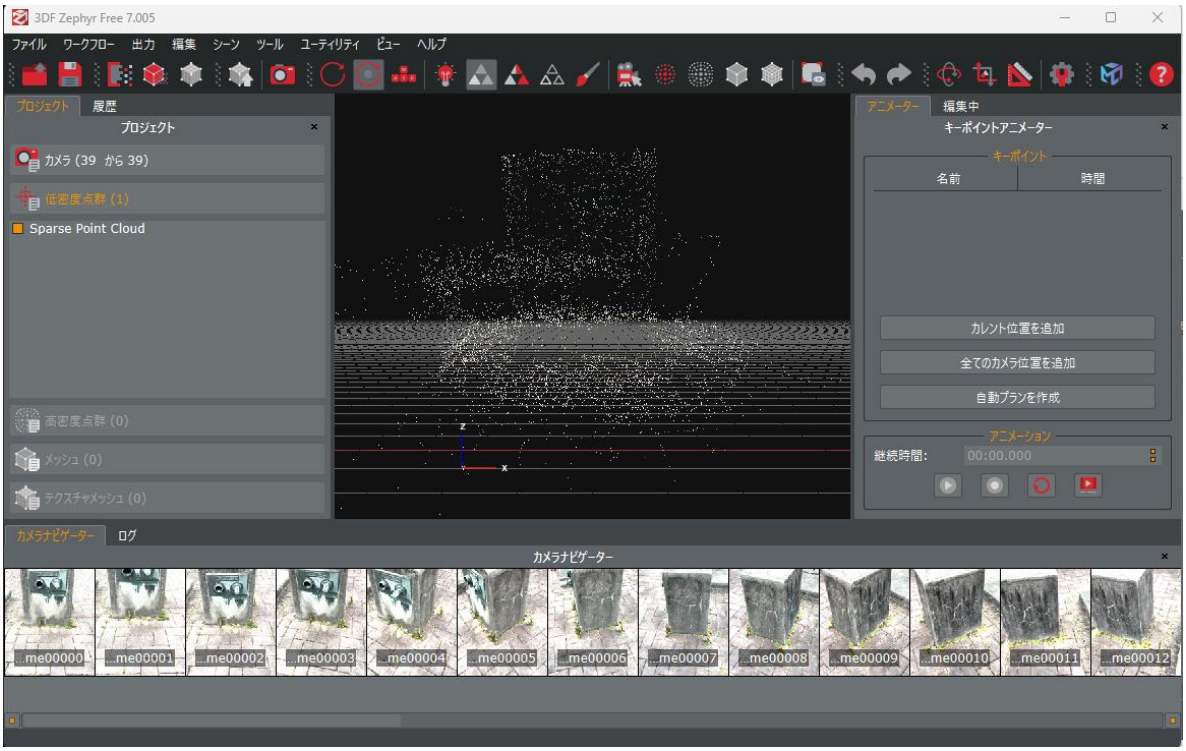
モデル化の設定を確認して「実行」をクリックします。



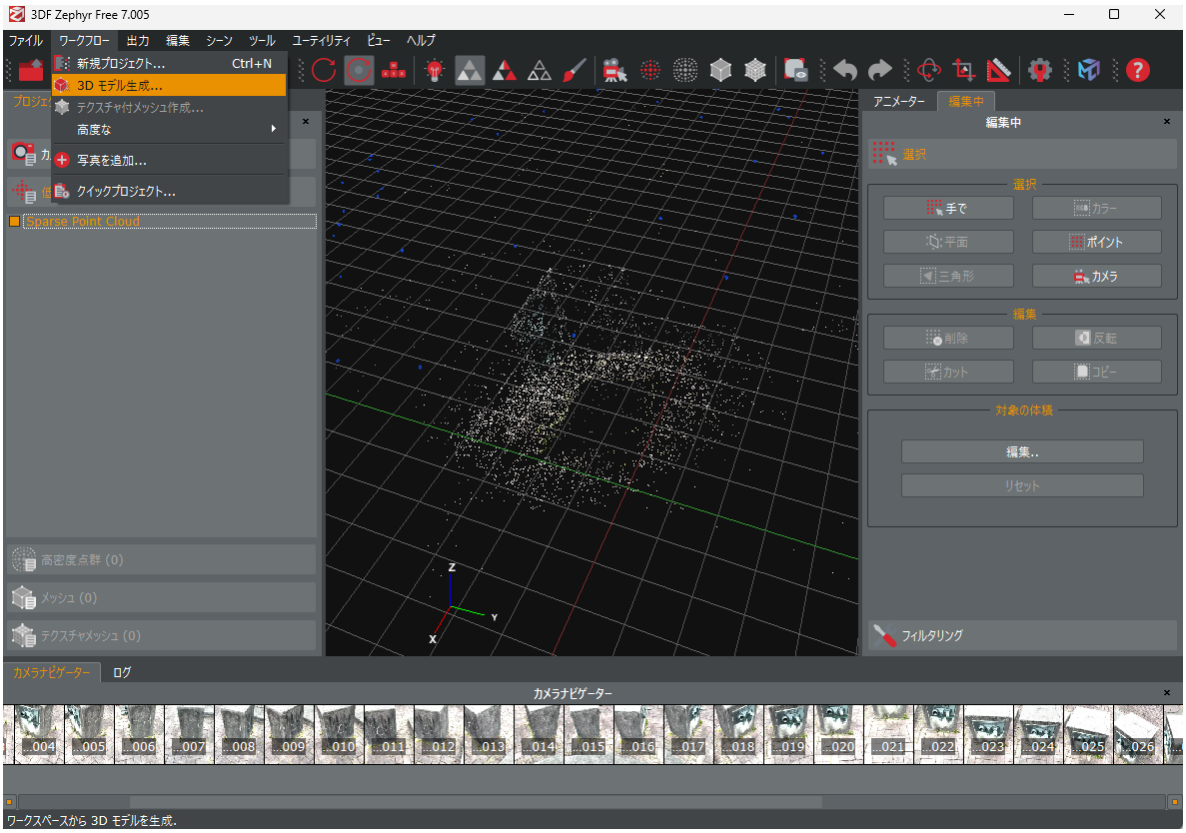
3 次元モデル作成が開始されます



3 次元モデル化に成功すると「モデル化に成功」ウィンドウが表示されるので「完了」ボタンをクリックします。



下の「カメラナビゲータ」のウィンドウに動画から切り出した画像が表示され、生成された低密度点群データが中央に表示されます



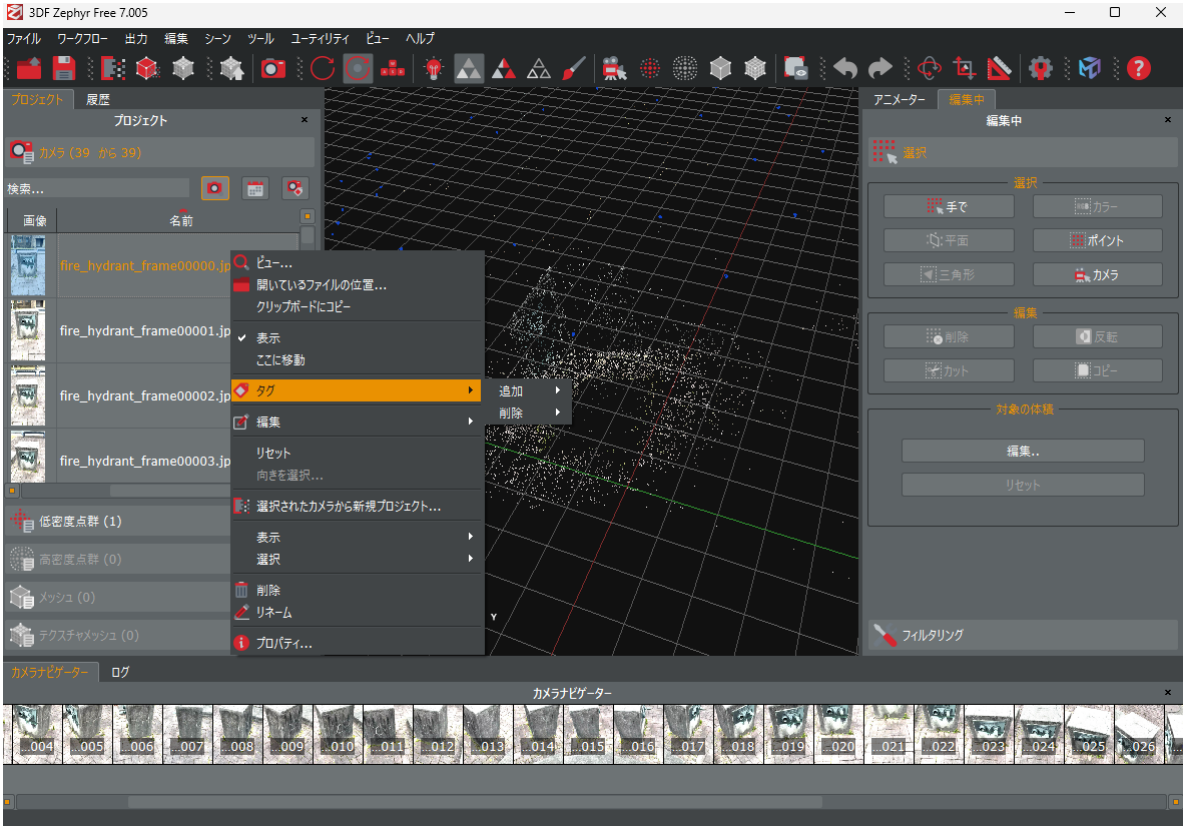
高密度点群データとメッシュデータを生成する為に、「3D モデル生成」メニューを選択します。

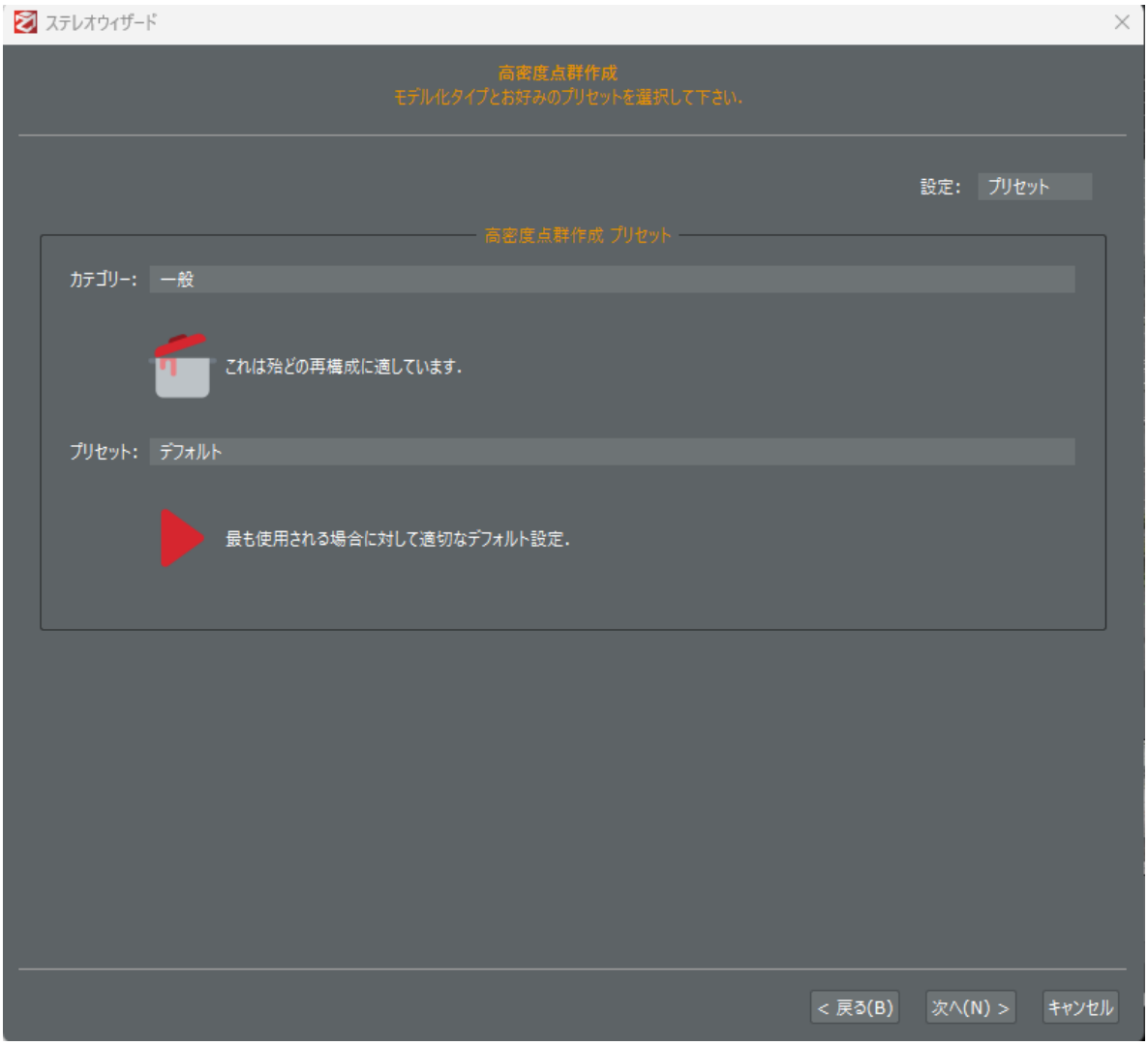
高密度点群データとメッシュデータを生成する為に、画像を選択して「次へ」ボタンをクリックします。

生成に使用する画像の選択は、今回は全ての画像データを使用するようにしてください。画像が見切れていたり、ターゲットの被写体以外の画像がある場合（ノイズがある場合）は、画像を除外することで、3D モデルの精度が向上する場合があります。

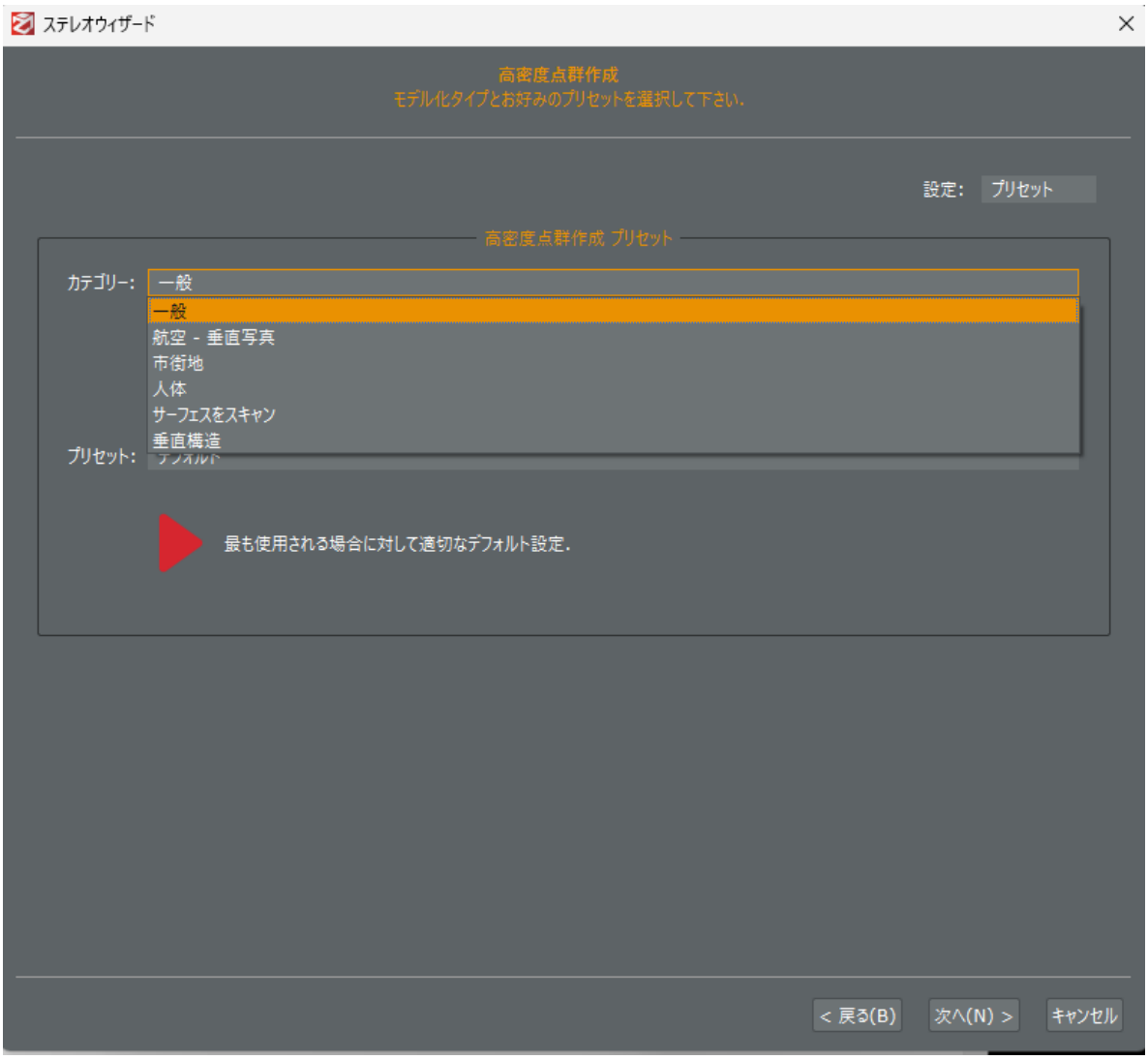
- 全てのカメラ：全ての画像データを使用します
- ワークスペースのカメラを選択：選択できません（グレー表示で無効）

- カメラを選択：全ての画像データから生成に使用する画像を選択します
- タグでカメラを選択：「タグ付け」された画像データを使用します。タグ付けは下記の例のように「カメラ」ウィンドウから右マウスメニューで設定をします





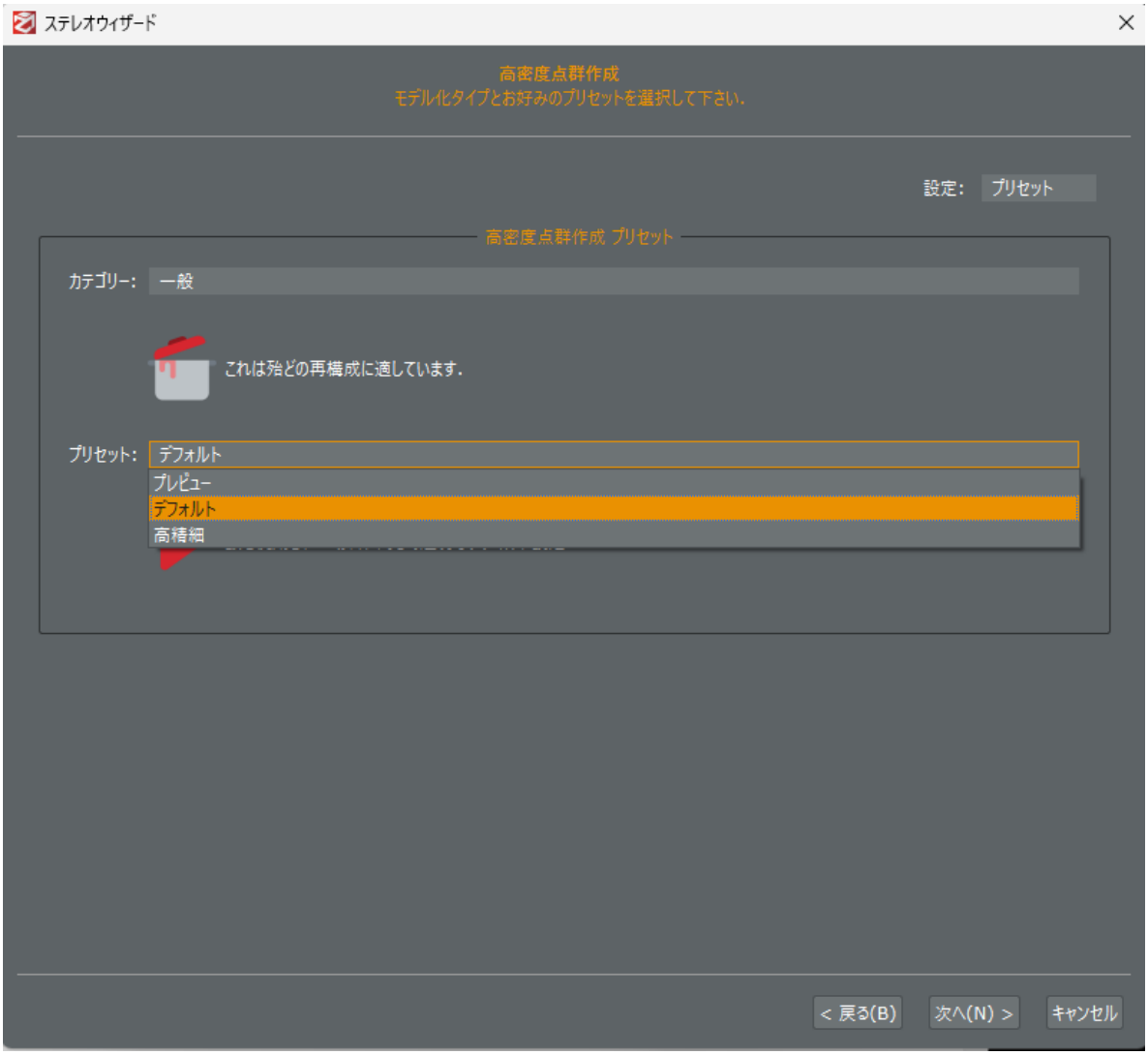
「高密度点群作成」では、モデル化する被写体の種類を設定します。通常はデフォルトの「カテゴリー：一般」「プリセット：デフォルト」を選択します。



被写体によってカテゴリーを変更して選択します。

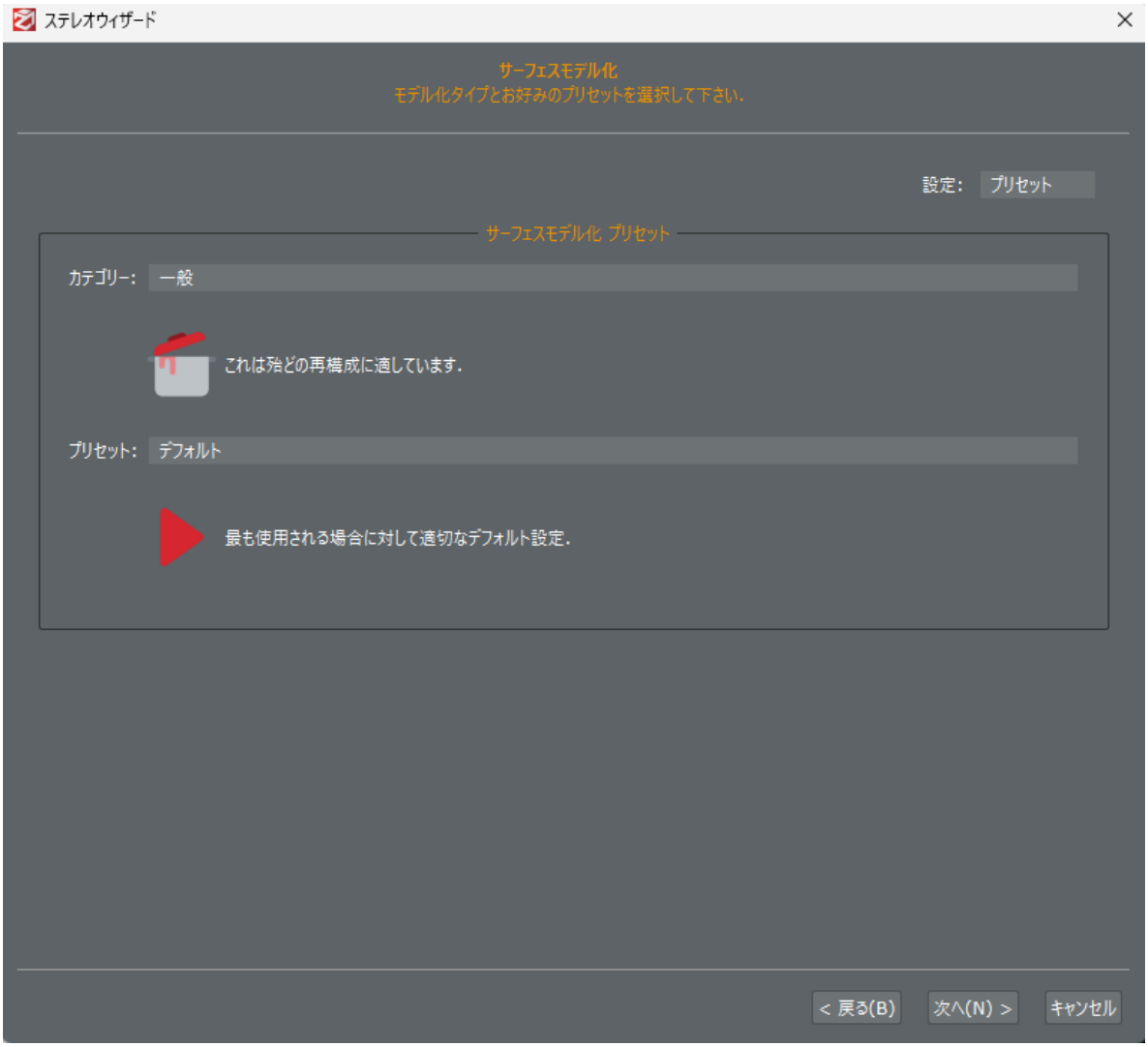
- 一般：一般的な場合に選択します
- 航空 - 垂直写真：ドローンや航空写真の場合に選択します
- 市街地：街の様々は被写体が混在している場合に選択します
- 人体：人が被写体の場合に選択します
- サーフェスをスキャン：近い位置から撮影した場合に選択します

- 垂直構造：通信塔やドローンで真上から撮った場合に選択します



プリセットを選択します

- プレビュー：作成済みの低密度点群データを使用するプリセット
- デフォルト：殆どの場合（ユースケース）に適したプリセット
- 高詳細：高密度の解像度の点群を作成するプリセット

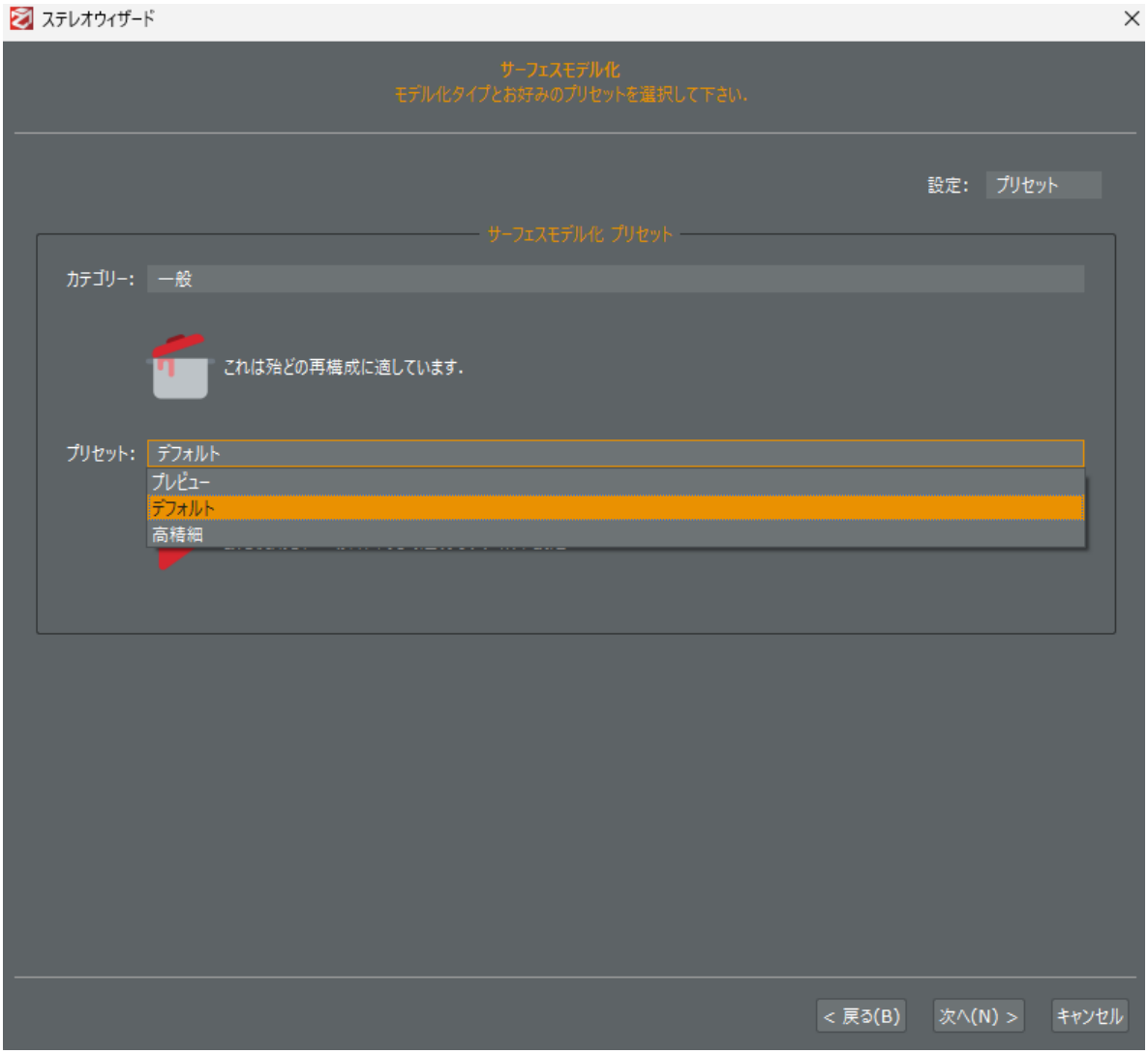


「サーフェスモデル化」は、モデル化する被写体の種類を設定します。通常はデフォルトの「カテゴリー：一般」「プリセット：デフォルト」を選択します。

被写体によってカテゴリーを変更して選択します。

- 一般：一般的な場合に選択します
- 航空 - 垂直写真：ドローンや航空写真の場合に選択します
- 市街地：街の様々は被写体が混在している場合に選択します
- 人体：人が被写体の場合に選択します
- サーフェスをスキャン：近い位置から撮影した場合に選択します

- 垂直構造：通信塔やドローンで真上から撮った場合に選択します



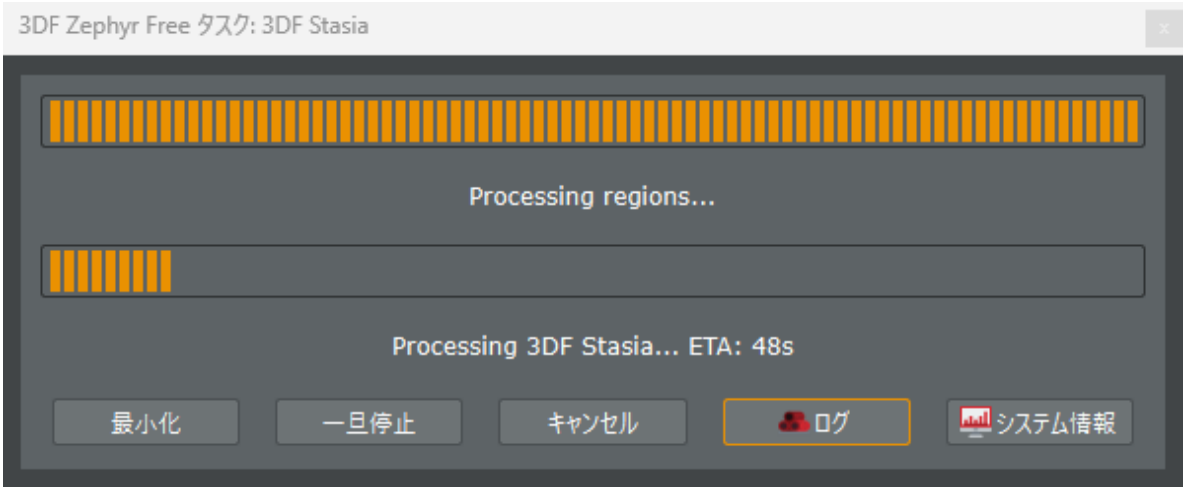
プリセットを選択します

- プレビュー：フォトコンシステンシーのフィルタ処理をしないプリセット
- デフォルト：殆どの場合（ユースケース）に適したプリセット

- 高詳細：フォトコンシステンシーのフィルタ処理の分解能を向上させて、より細かくメッシュを作成する設定のプリセット

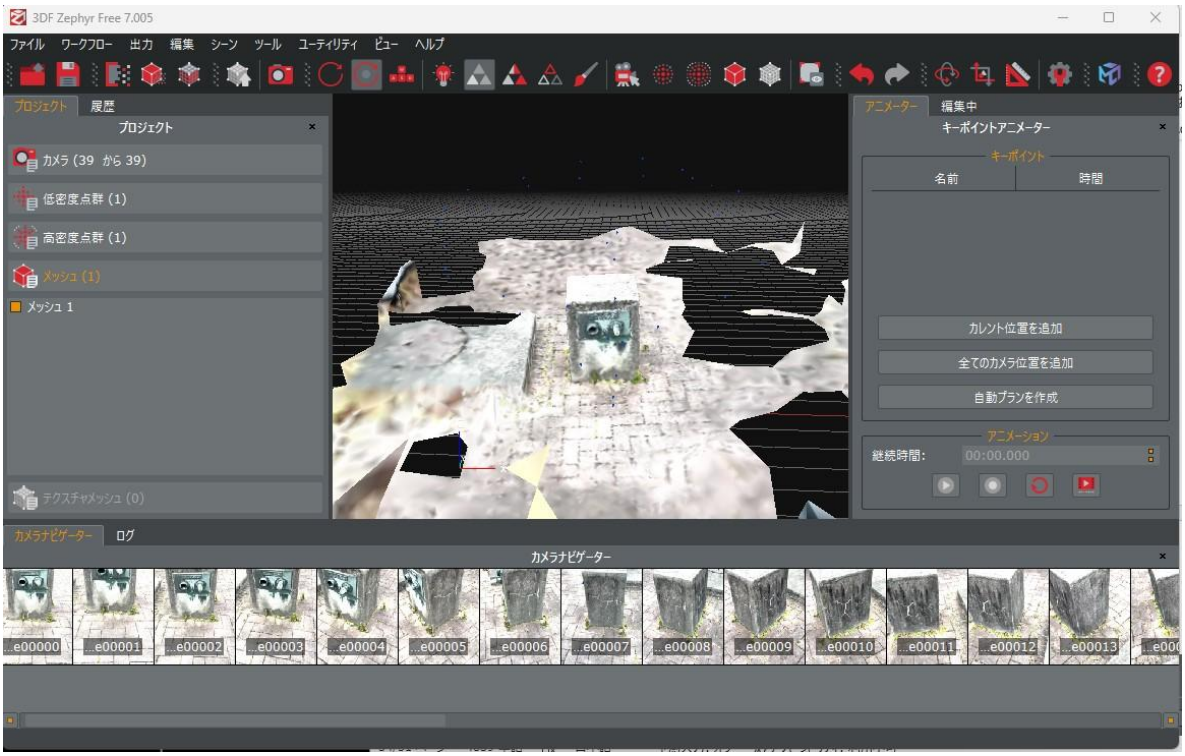
※ フォトコンシステンシー(Photo-consistency)フィルタは、リアルでシャープなエッジを実現するためのフィルタ処理

高密度点群データとメッシュデータの設定を確認して「Run（実行）」をクリックします。

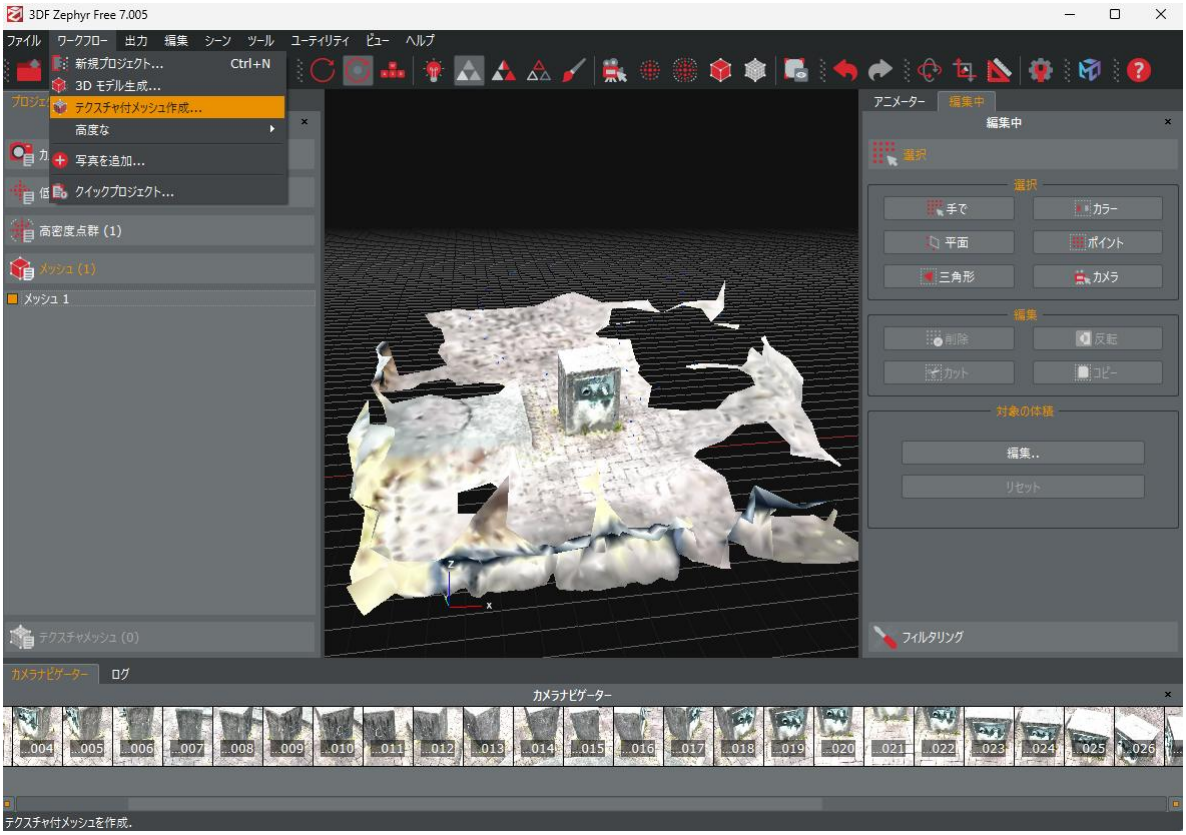


高密度点群データとメッシュデータの 3 次元モデル作成が開始されます

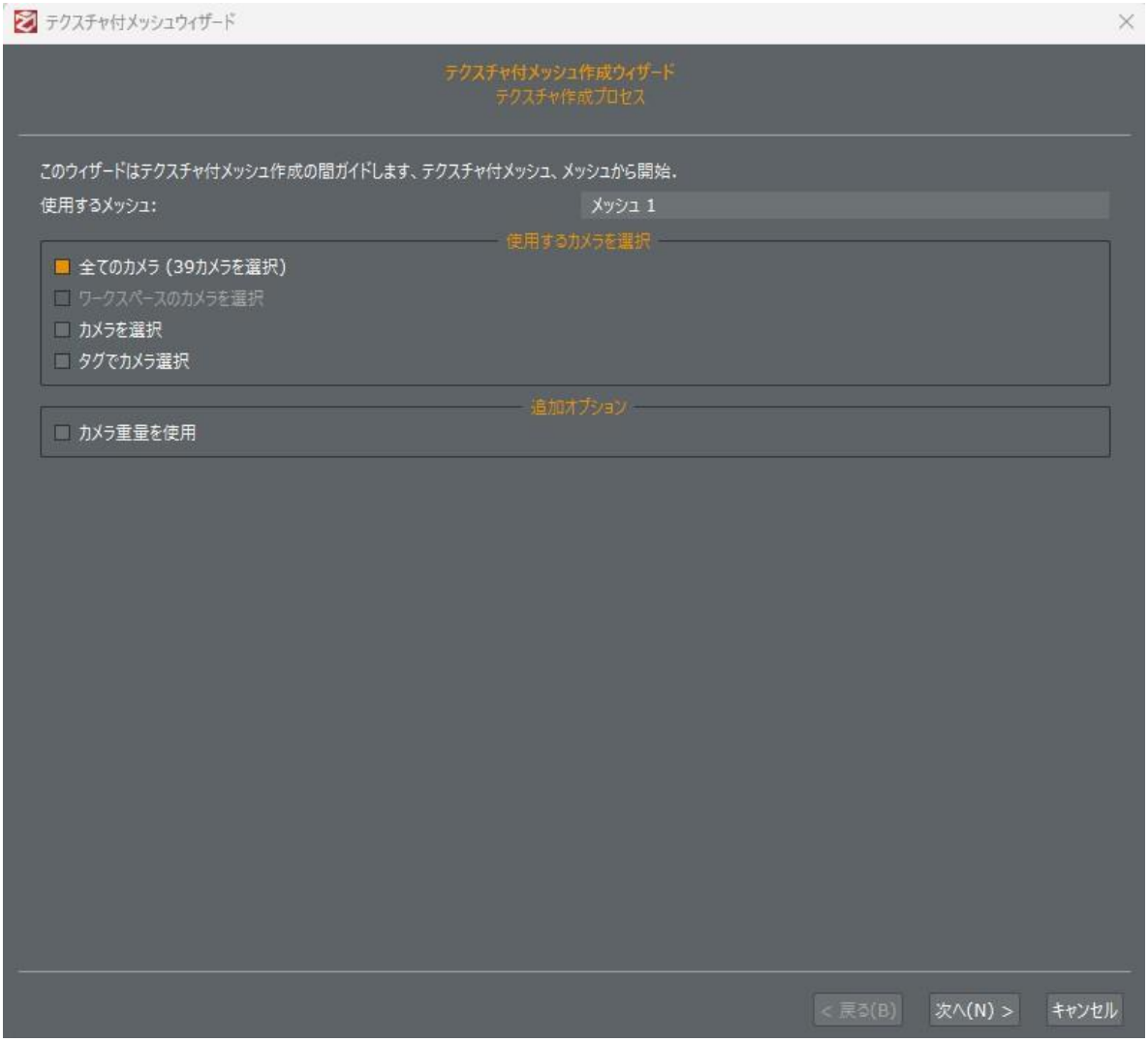
3 次元モデル化に成功すると「3D モデル生成に成功」ウィンドウが表示されるので「完了」ボタンをクリックします。



生成された高密度点群データとメッシュデータが表示されます



テクスチャ付きのメッシュデータを生成する為に、「テクスチャ付きのメッシュ作成」メニューを選択します。テクスチャとは 3 次元モデルの表面に画像(テクスチャ)データを張り付けてものです。



テクスチャ付きのメッシュデータを生成する為に、使用するメッシュで、作成したメッシュデータ「メッシュ１」選択して「次へ」ボタンをクリックします。

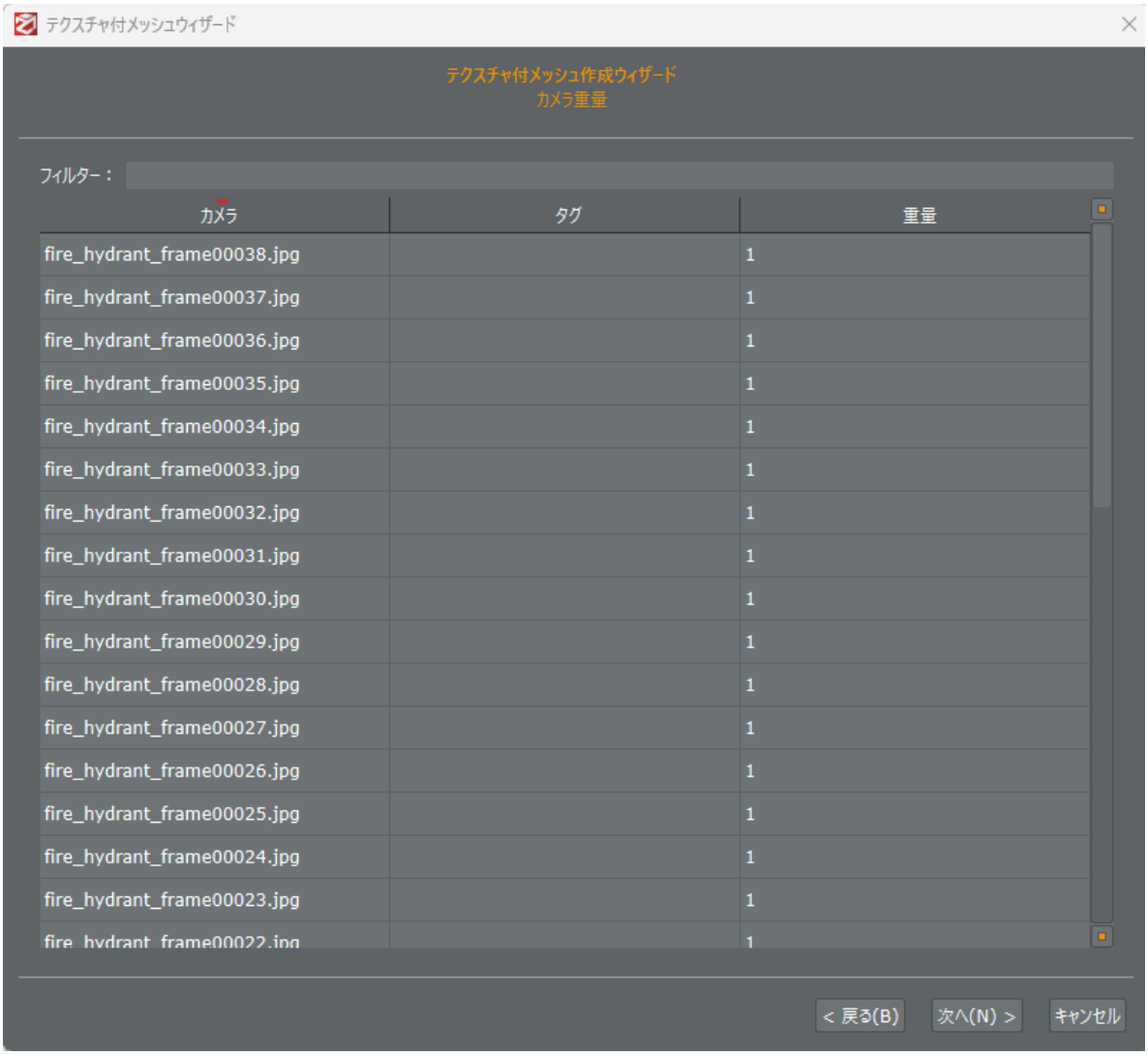
テクスチャに使用する画像の選択は、今回は全ての画像データを使用するようにしてください。画像が見切れていたり、ターゲットの被写体以外の画像がある場合（ノイズがある場合）は、画像を除外することで、メッシュデータに張り付けるテクスチャの精度が向上する場合があります。

- 全てのカメラ：全ての画像データを使用します

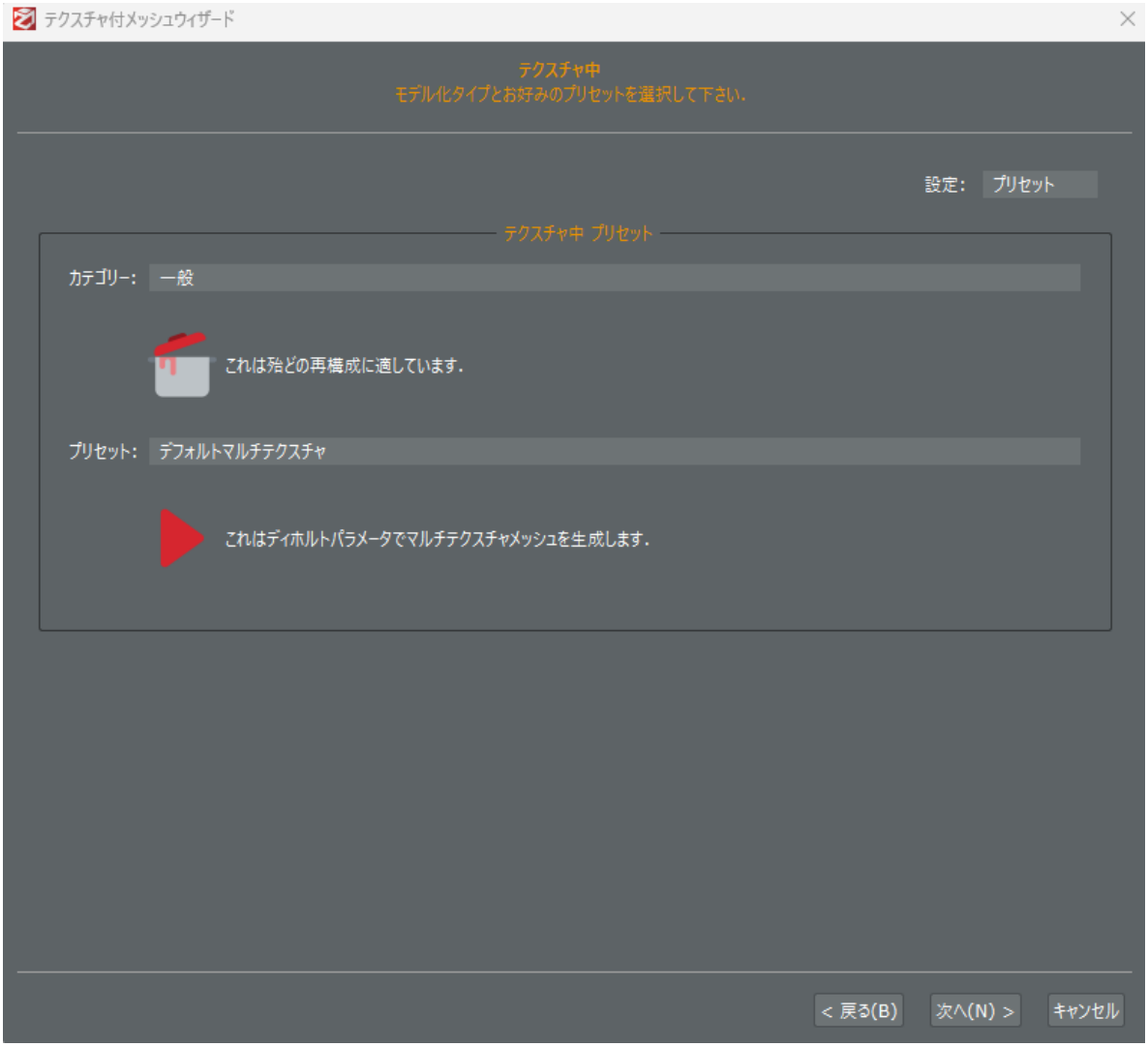
- ワークスペースのカメラを選択：ここでは選択できません
- カメラを選択：全ての画像データから生成に使用する画像を選択します
- タグでカメラを選択：「タグ付け」された画像データを使用します

追加オプション：

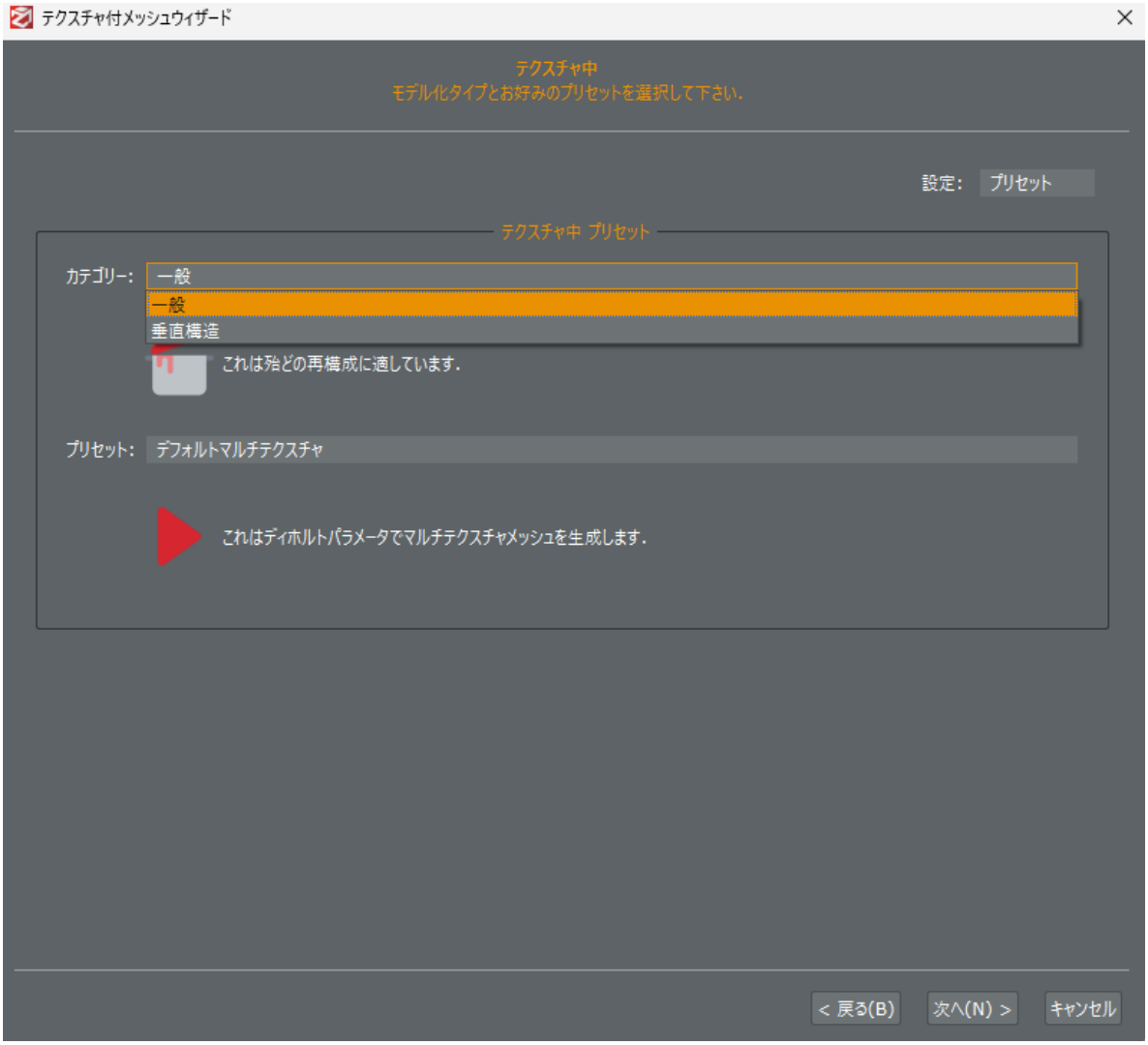
- カメラ重量を使用：画像データの重み付け値（0.1～1.0）を使用する場合に ON にします。
  重み付け値が大きいほど画像データがモデル生成のテクスチャに使用される優先順位が
  上がります。



各画像データの重み付け値（0.1～1.0）を個別に設定します。デフォルトでは全て「1」の値になってます。全て「1」の値の場合は優先順位はないことになります。

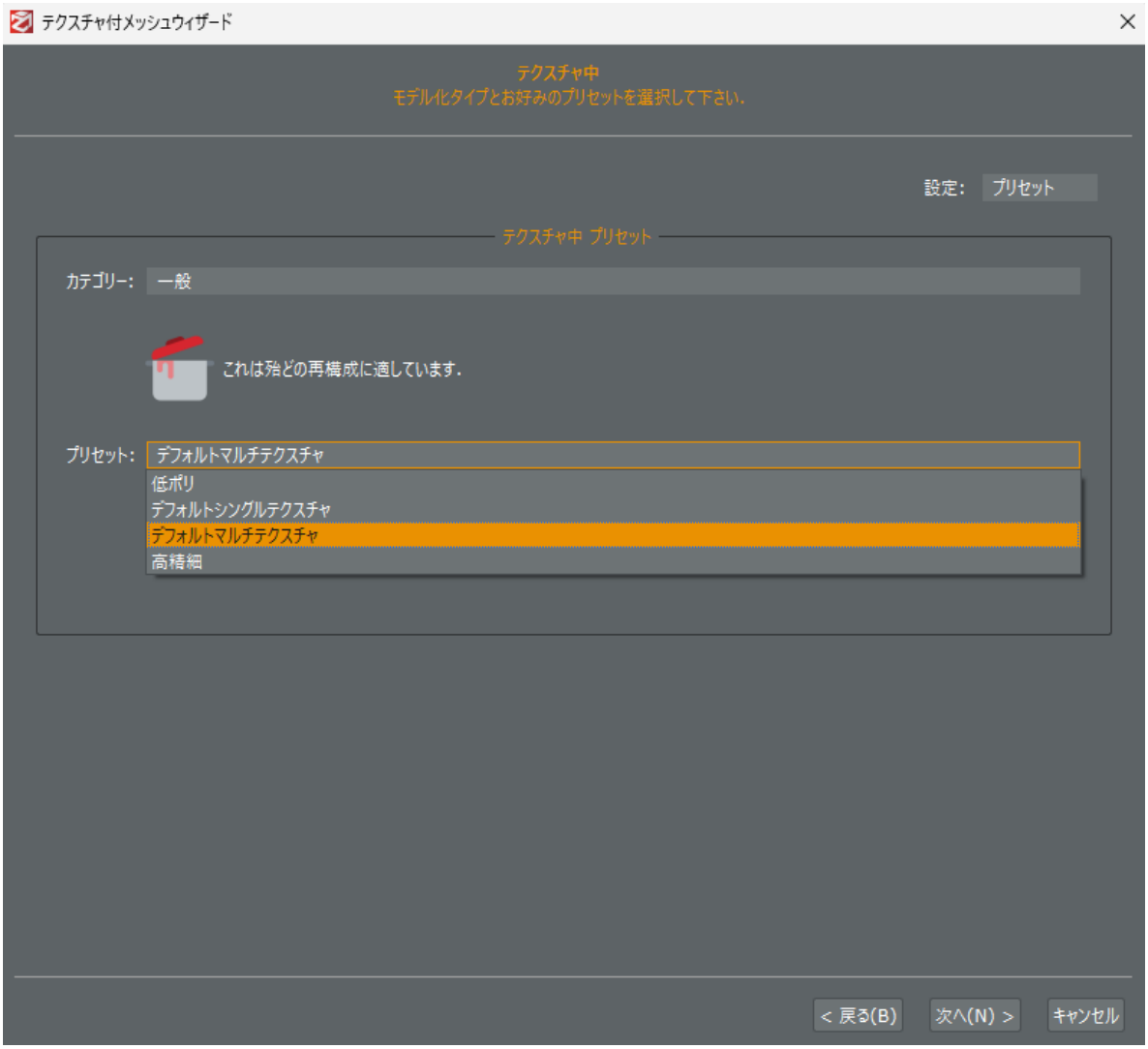


「テクスチャ中」は、モデル化する被写体の種類を設定します。通常はデフォルトの「カテゴリー：一般」「プリセット：デフォルトマルチテクスチャ」を選択します。



被写体によってカテゴリーを変更して選択します。

- 一般：一般的な場合に選択します
- 垂直構造：真上から撮った場合に選択します



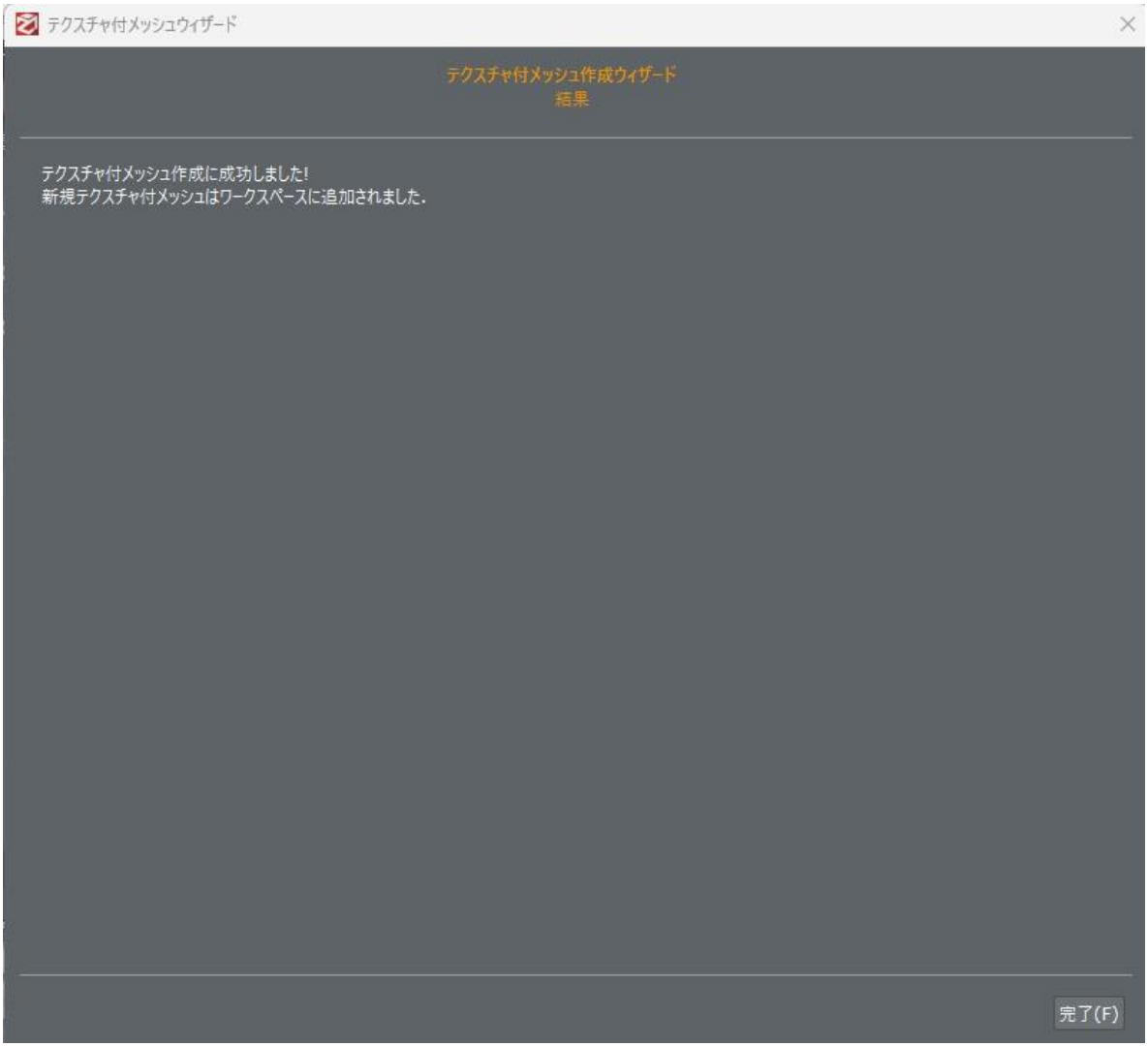
プリセットを選択します

- 低ポリ：単一面メッシュが生成され、ファイルサイズが小さくなるプリセット
- デフォルトシングルテクスチャ：デフォルトパラメータで単一面のメッシュが生成されるプリセット

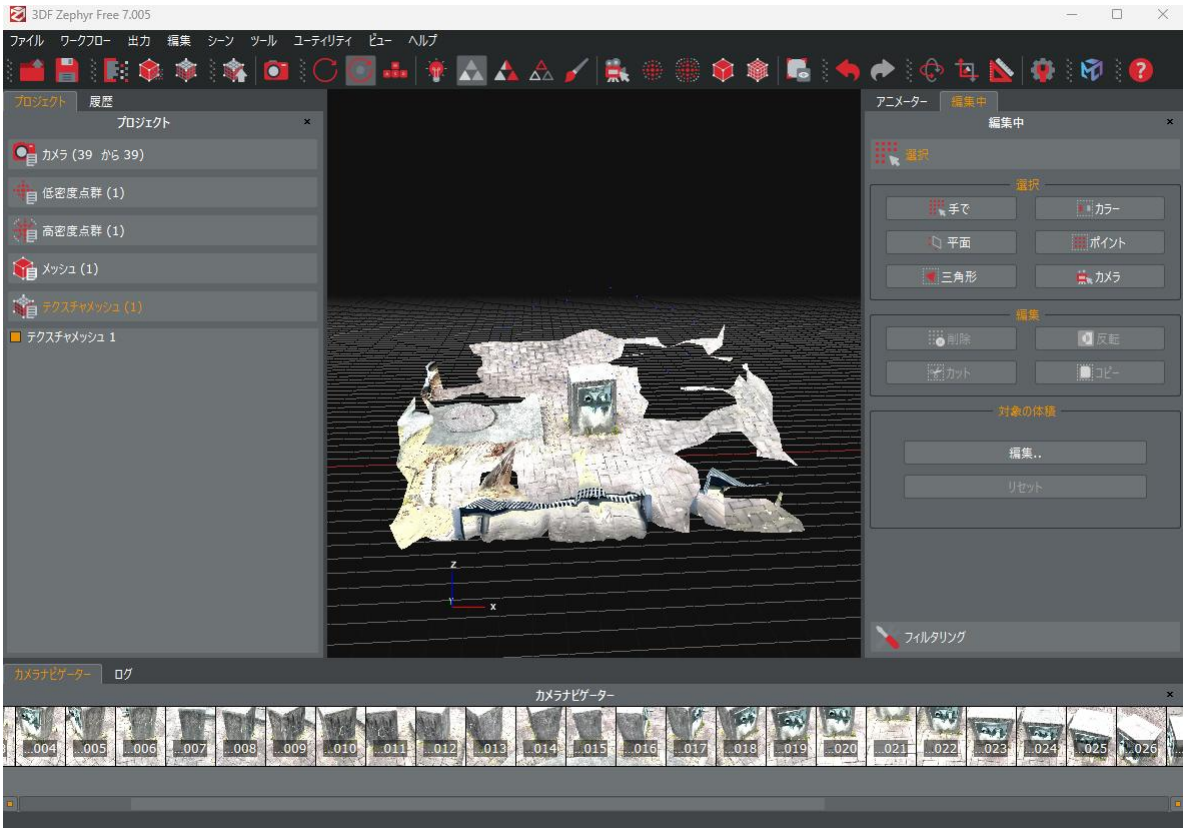
- デフォルトマルチテクスチャ：デフォルトパラメータで複合面のメッシュが生成されるプリセット
- 高詳細：高解像度の複合面のッシュが生成されるプリセット

テクスチャ付きのメッシュデータのモデル化の設定を確認して「実行」をクリックします。

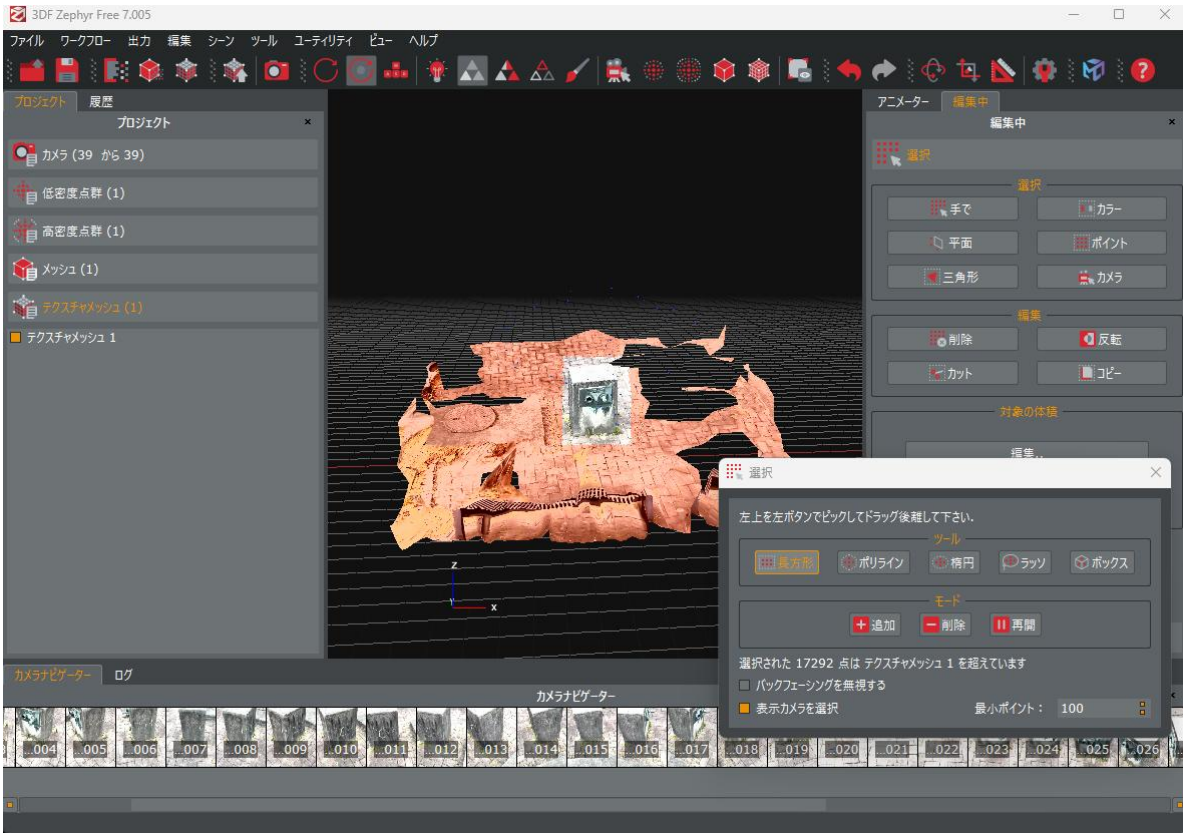
テクスチャ付きのメッシュデータの 3 次元モデル作成が開始されます



テクスチャ付きのメッシュデータの生成に成功すると「テクスチャ付きメッシュ作成ウィザード結果」ウィンドウが表示されるので「完了」ボタンをクリックします。



生成されたテクスチャ付きのメッシュデータが表示されます



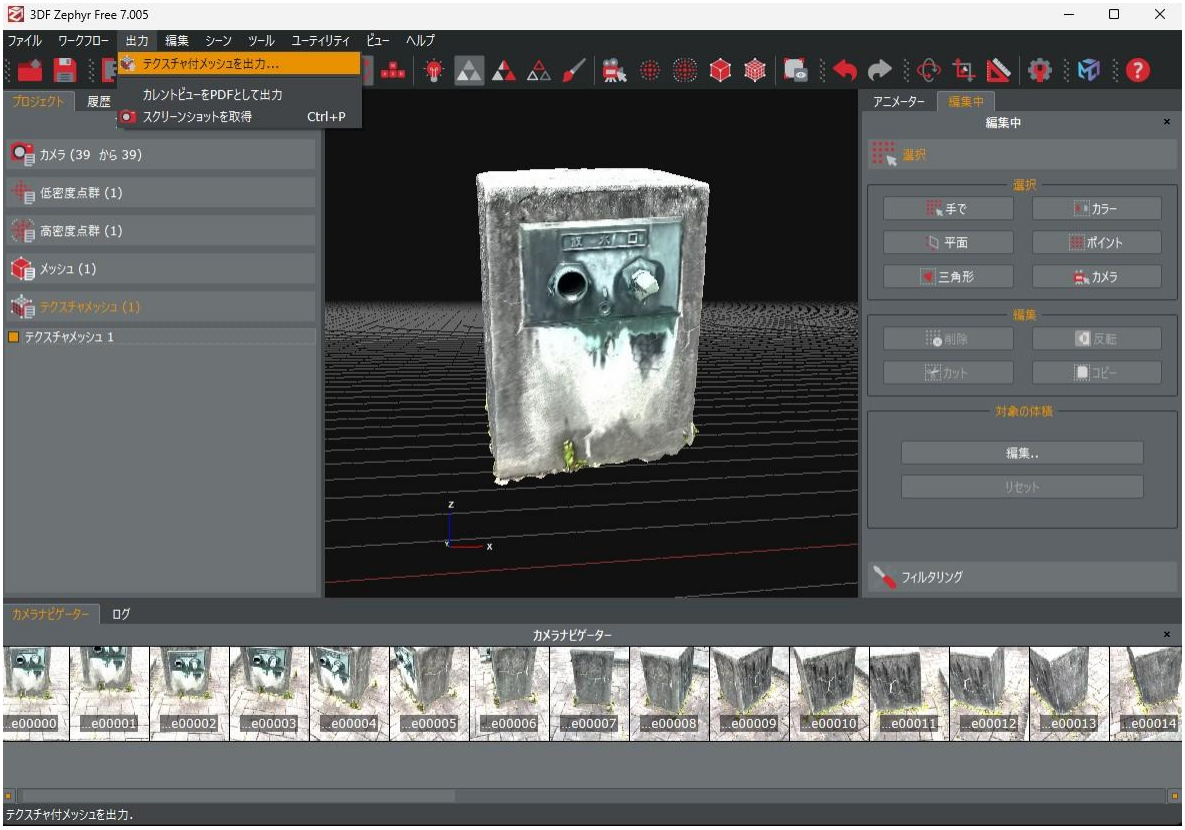
生成された 3 次元モデルが表示された後は、右端にある編集ウィンドウで余分な部分を削除します。

編集ウィンドウの役割

選択 ：画像の選択方法

編集：削除やコピーをします。反転ボタンで選択が反転します

詳細な編集方法は、【フォトグラメトリ】3DF Zephyr を使ってモデルのメッシュを綺麗にする方法（https://styly.cc/ja/tips/3dfzephyr_processing/）等で編集方法を参考にしてください。



最後に、生成したテクスチャ付きのメッシュデータの 3 次元モデルをファイルに出力します。

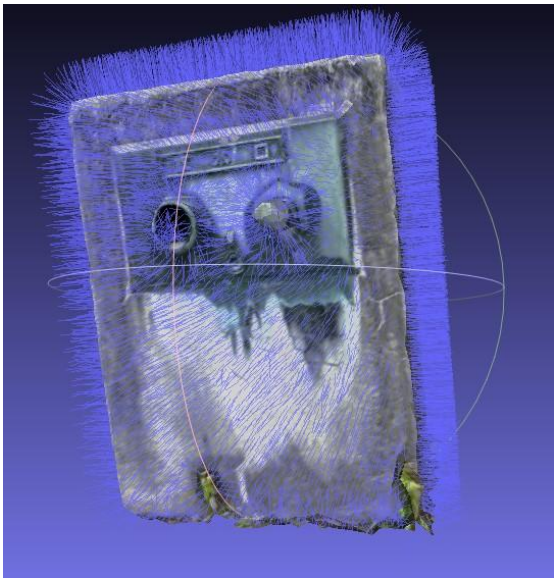
メニューから「出力」→「テクスチャ付きメッシュを出力...」を選択します

「テクスチャ付きメッシュを出力」ダイアログが表示されるので、出力フォーマットを設定して「出力」ボタンをクリックして、テクスチャ付き PLY フォーマットの 3 次元モデルファイルを保存します。

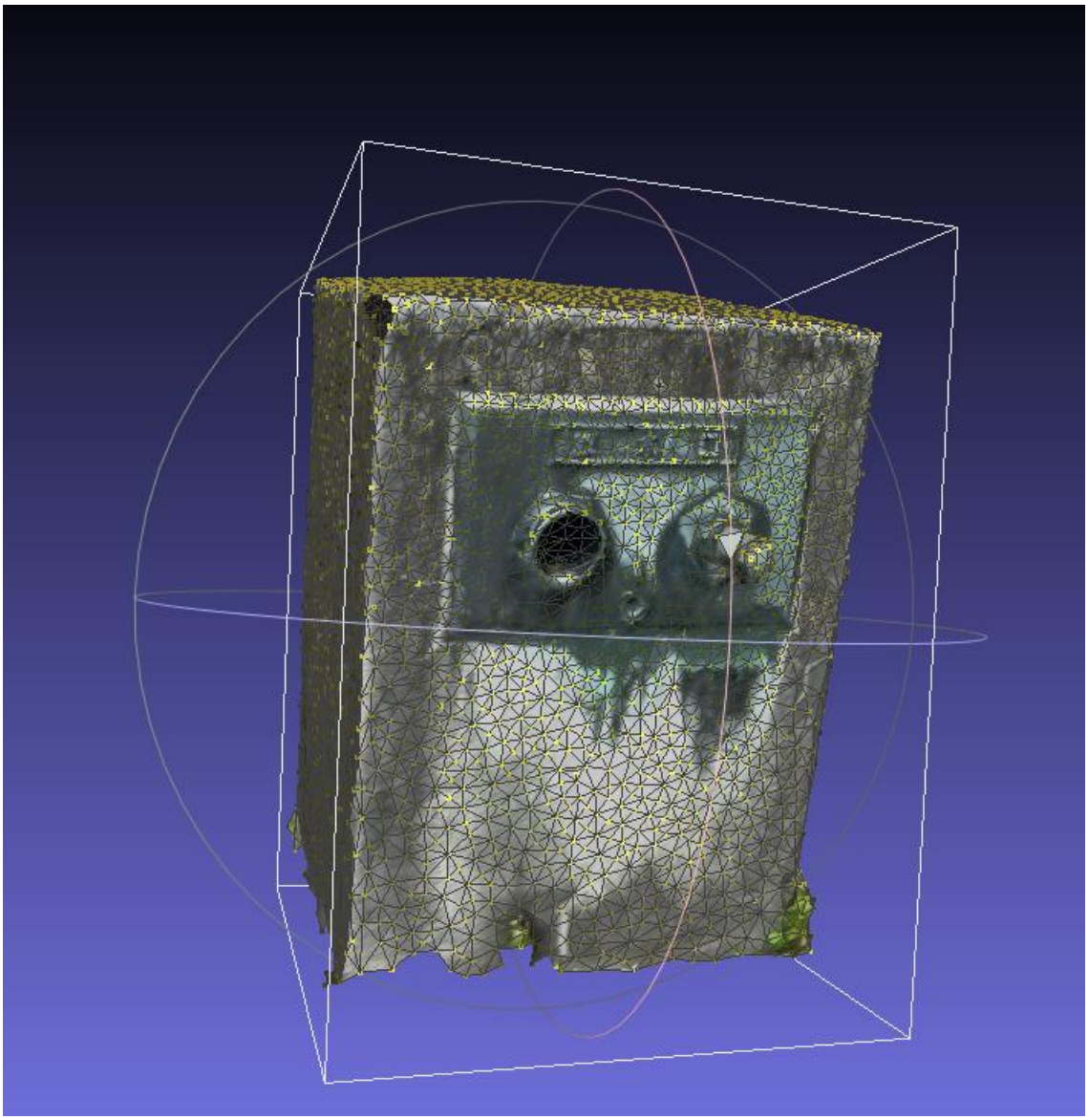
出力フォーマット

- フォーマットを出力：ply を選択
- 法線を出力：ON　法線を出力します
- バイナリエンコードを使用：ON　データをバイナリデータで出力します

※ 法線：法線は面の向きを定義するもので、面に対し垂直に線が伸びたものです。この線が出ている面が「表の面」になります。



※ バイナリデータ：主にコンピュータが理解するために記述されたデータ形式で人間が読めない形式です。ON にしなければ、テキストデータ（文字データ）で保存されるので、人間が読める形式で保存されます

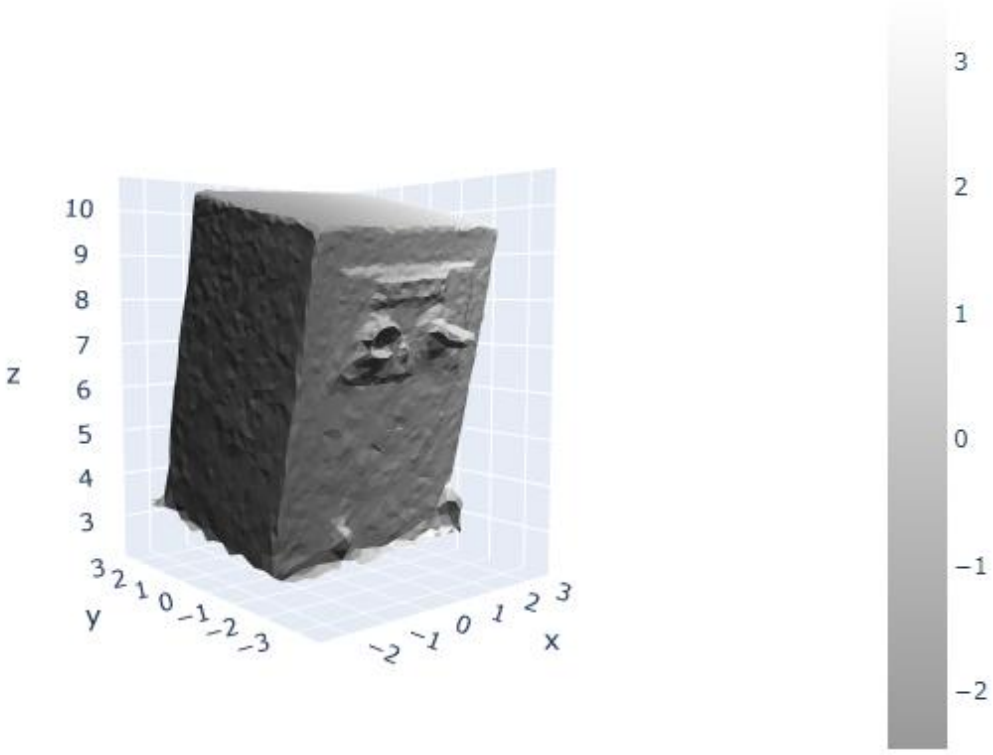


図：テクスチャ付き PLY フォーマットの 3 次元モデルファイル: メッシュと頂点を表示

3 次元モデルを表示

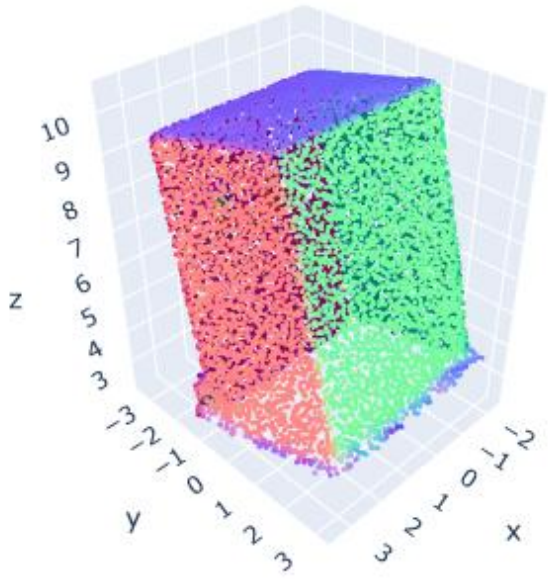
作成した 3 次元モデルファイルを Google Colab で表示していきます。Open3D (https://github.com/isl-org/Open3D/releases) 0.16.0 バージョンから、Google Colab でも Plotly で可視化できるようになったので 、この機能を利用します。

```
# 3 次元モデル読み込み
file_path = "/content/gdrive/MyDrive/dataset/image_in/fire_hydrant.ply
"    # ファイルパスを設定
fp_mesh = o3d.io.read_triangle_mesh(file_path)   # メッシュデータ読み込み
o3d.visualization.draw_plotly([fp_mesh])   # メッシュデータ表示
```



まずは、PLY ファイルを読み込んでメッシュデータを表示します。

```
# メッシュポリゴンから点群への変換
fp_point = fp_mesh.sample_points_uniformly(number_of_points=10000)   #
メッシュからポイントデータ作成
o3d.visualization.draw_plotly([fp_point]) # ポイントデータ表示
```



次に、メッシュデータからポイントデータに変換して表示してみます。

3 次元モデルを加工

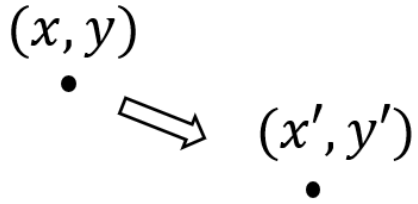
3 次元モデルを画像変換で「拡大/縮小」「回転」「平行移動」に加工してみます。

画像変換の「拡大/縮小」「回転」「平行移動」は三角関数と行列に計算が使われています。

例えば、2 次元データで原点を中心に θ 回転させるとした場合は下記となります。

$$\begin{cases} x' = x\cos\theta - y\sin\theta \\ y' = x\sin\theta + y\cos\theta \end{cases}$$

この計算から、座標(x, y)を座標(x', y')に変換します



行列は、連立１次方程式の解法の工夫から行列という考えが生まれベクトルや行列式とともに線形代数という数学の分野になっています。 簡単に言えば数値を縦方向と横方向に状に並べて表現をしたもので、複雑な数式をスッキリ書くことができます。

行列の例として、座標変換を連立方程式にすると次式になるとします。

$$\begin{cases} 2x + y + 3z = x_1 \\ x - y + z = y_1 \\ x + 2y - z = z_1 \end{cases}$$

同じものを行列式で表現すると次のように表記できます

$$\begin{pmatrix} 2 & 1 & 3 \\ 1 & -1 & 1 \\ 1 & 2 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$$

つまり、こんな意味になります

(変換のルール(法則))(元のx, y, z座標) ＝ (変換後のx, y, z座標)

Open3D ライブラリでも、画像の各画素(ピクセル)の点の位置を変換するのみ三角関数と行列を使用しています。

では、実際に画像変換をしていきます。

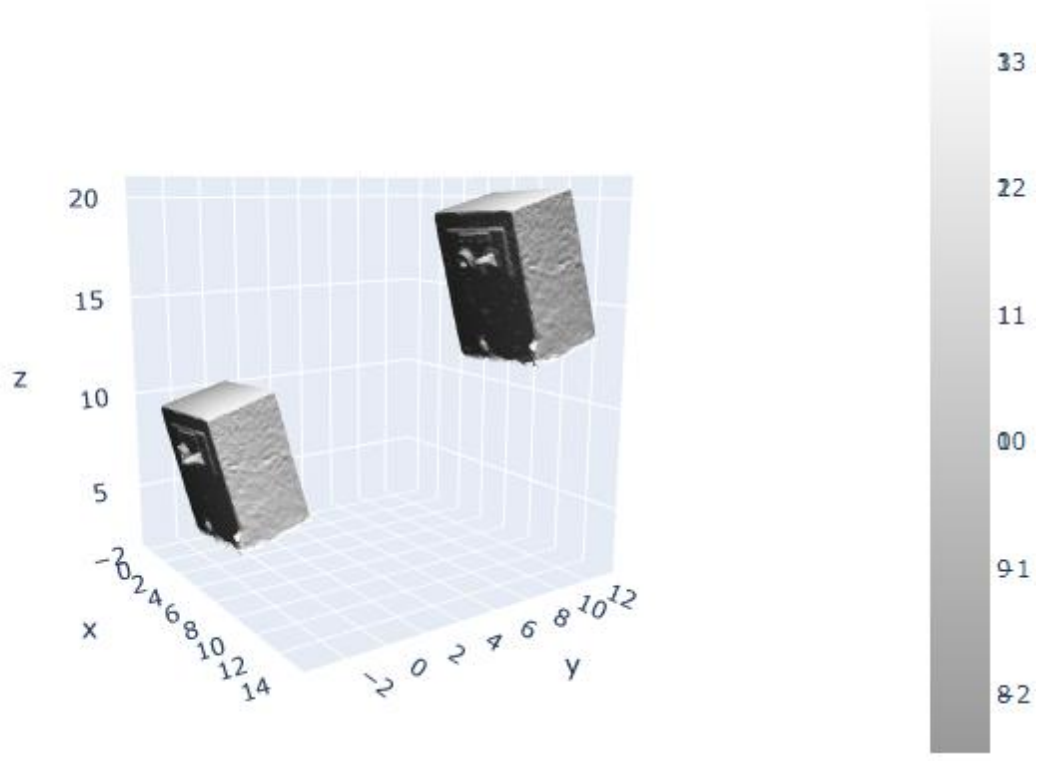
移動

データを移動させてみます。3 次元モデルを形成している（X,Y.Z）の座標点を、それぞれ(10,10,10）移動させます

表示は比較しやすいように、変換前と変換後の２つを表示させています。

```
# 移動:並進移動
def translate(mesh):
    mesh_t = copy.deepcopy(mesh).translate([10, 10, 10])  # データをコピーして、X,Y,Z の移動量を設定
    o3d.visualization.draw_plotly([mesh, mesh_t])  # 元データと移動したデータを並べて表示

translate(fp_mesh)  # メッシュデータを移動表示
```



回転

データを回転させます

回転は軸角度からデータを回転させます。表示はメッシュデータを表示させます

例は x 軸回転に指定なので、3 次元の x 軸周りの 3x3 回転行列との積で座標を求めます

| 1　0　　0　|
| 0 cosθ　-sinθ |
| 0 sinθ　 cosθ |

※ 回転行列 (rotation matrix)

https://w3e.kanazawa-it.ac.jp/math/category/gyouretu/senkeidaisu/henkan-tex.cgi?target=/math/category/gyouretu/senkeidaisu/rotation_matrix.html

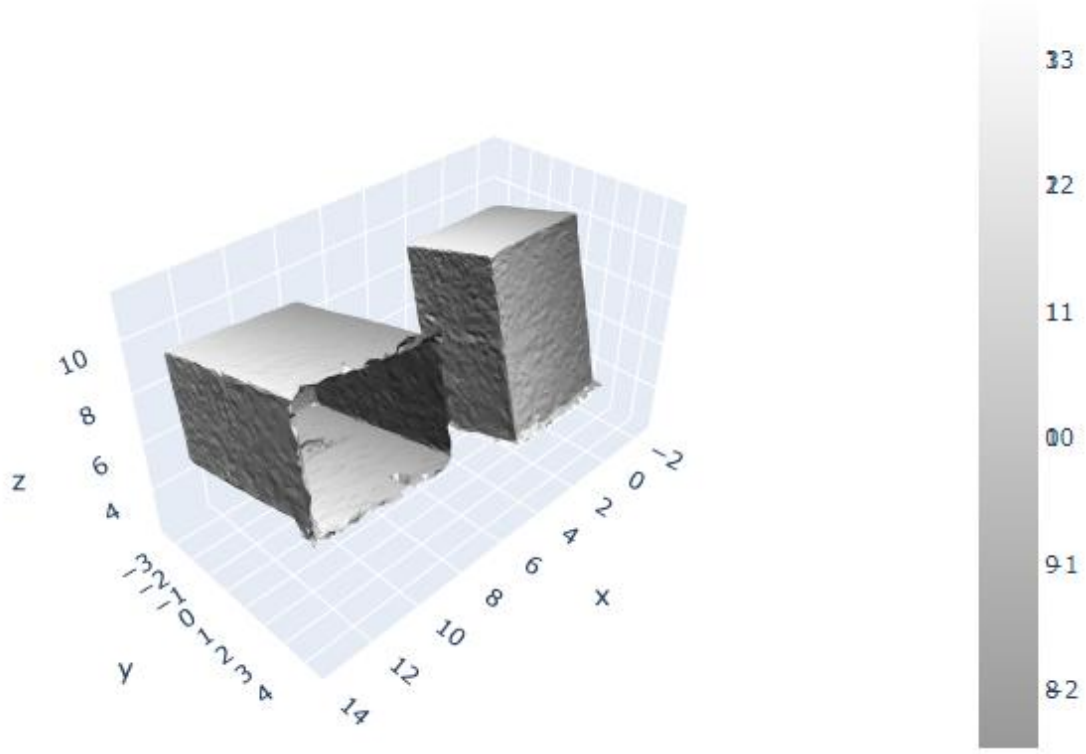
※ 軸角度：X 軸、Y 軸、Z 軸を始点に回る角度です

```
# 回転:軸角度からデータを回転
# メッシュデータを表示
def rotate_axis(mesh):
    mesh_r = copy.deepcopy(mesh).translate((10, 0, 0))  # 並べ表示用するために、移動量を指定してコピー
    rotation_matrix = o3d.geometry.get_rotation_matrix_from_axis_angle([np.radians(90), 0, 0]) # 回転行列に変換
    mesh_r.rotate(rotation_matrix) # データ回転
    o3d.visualization.draw_plotly([mesh, mesh_r]) # 元データと回転したデータを表示

rotate_axis(fp_mesh)  # メッシュデータを回転表示
```

回転行列を表示

```
[[ 1.000000e+00  0.000000e+00  0.000000e+00]
 [ 0.000000e+00  6.123234e-17 -1.000000e+00]
 [ 0.000000e+00  1.000000e+00  6.123234e-17]]
```



次に、回転方法を変えてはオイラー角を使用してデータを回転させます。表示はポイントデータを表示させて、変換前と変換後で色を変えてみます。

※ オイラー角：回転の組み合わせで、回転させた物体の座標系の軸で回転をすること。例えば、「X 軸周りに 30 度，次に回転した物体の Y 軸を中心に 45 度，次に物体の Z 軸周りに 90 度」という形です https://ja.wikipedia.org/wiki/オイラー角#/media/ファイル:Euler2a.gif

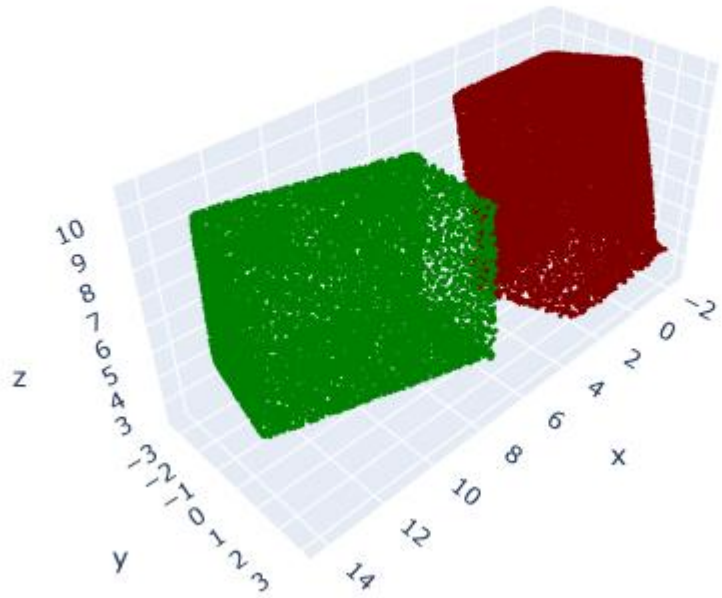
```
# 回転:オイラー角を使用してデータを回転
# ポイントデータを表示
def rotate_xyz(mesh):
    mesh_r = copy.deepcopy(mesh).translate((10, 0, 0))  # 並べ表示用するために、移動量を指定してコピー
    rotation_matrix = o3d.geometry.get_rotation_matrix_from_xyz((0, np.pi / 2, 0))  # 回転行列に変換
    mesh_r.rotate(rotation_matrix) # 回転
```

```
    mesh.paint_uniform_color([0.5, 0.0, 0.0])       # 元データを赤表示
    mesh_r.paint_uniform_color([0.0, 0.5, 0.0])     # 回転データを緑表示
    o3d.visualization.draw_plotly([mesh, mesh_r])   # 元データと回転したデータを表示

rotate_xyz(fp_point)   # ポイントデータを回転表示
```

回転行列を表示

```
[[ 6.123234e-17  0.000000e+00  1.000000e+00]
 [ 0.000000e+00  1.000000e+00  0.000000e+00]
 [-1.000000e+00  0.000000e+00  6.123234e-17]]
```



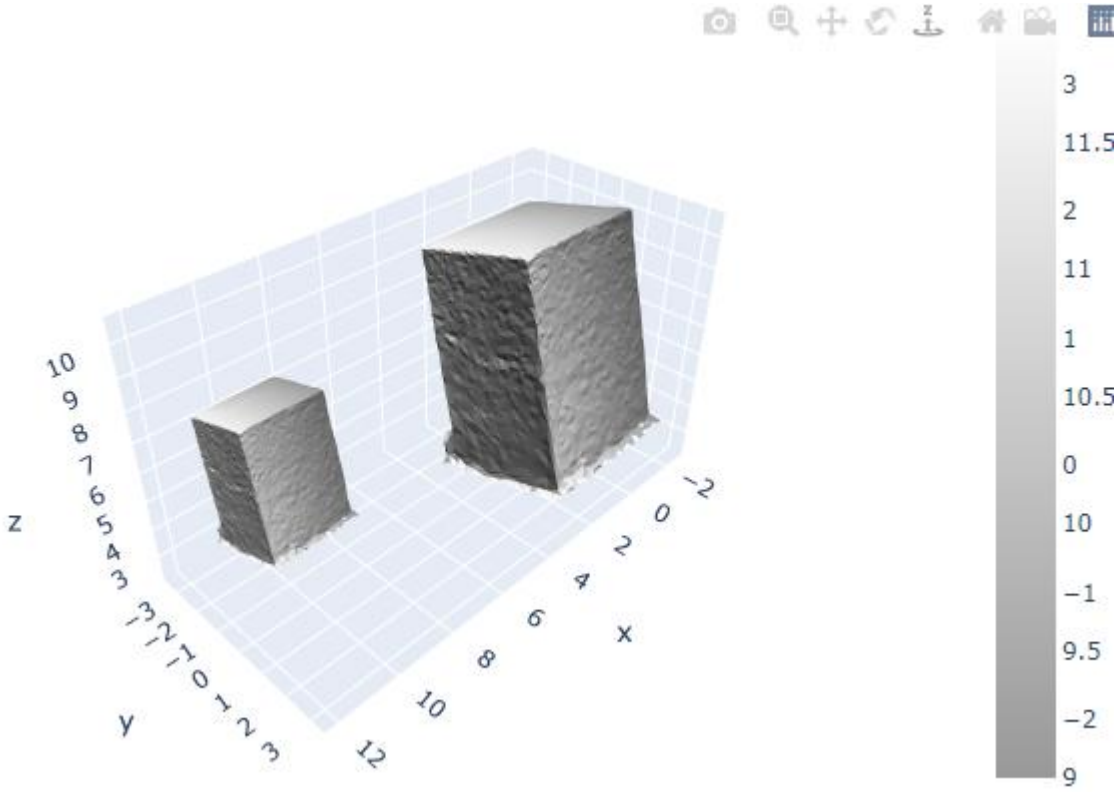
スケール

データをスケール（拡大・縮小）させます。

元の３D モデル（下の図で右側）を、コピーして、 x 軸方向に＋10 移動させた位置で、立体の中心を基点にして５０パーセント縮小して表示させます。

```
# スケール: データを拡大・縮小
def scale(mesh):
    mesh_s = copy.deepcopy(mesh).translate((10, 0, 0))  # 並べ表示用するために、移動量を指定してコピー
    mesh_s.scale(0.5, center=mesh_s.get_center())  # メッシュデータの中心を軸に、スケールデータ作成
    o3d.visualization.draw_plotly([mesh, mesh_s])  # 元データとスケールしたデータを表示

scale(fp_mesh)  # メッシュデータをスケール表示
```



転換［変換］

同次変換行列を利用して、回転と移動を同時に実行させます。

指定は x 軸回転の同次変換なので、下記の 4x4 の同次変換行列の積となります

| 1　　0　　0　　x 移動 |

| 0　cosθ　-sinθ　y 移動 |

| 0　sinθ　cosθ　z 移動 |

| 0　　0　　0　　1 |

※　同時変換配列： 位置と回転とを一つの行列にまとめ上げ 4x4 の行列にしたもの.

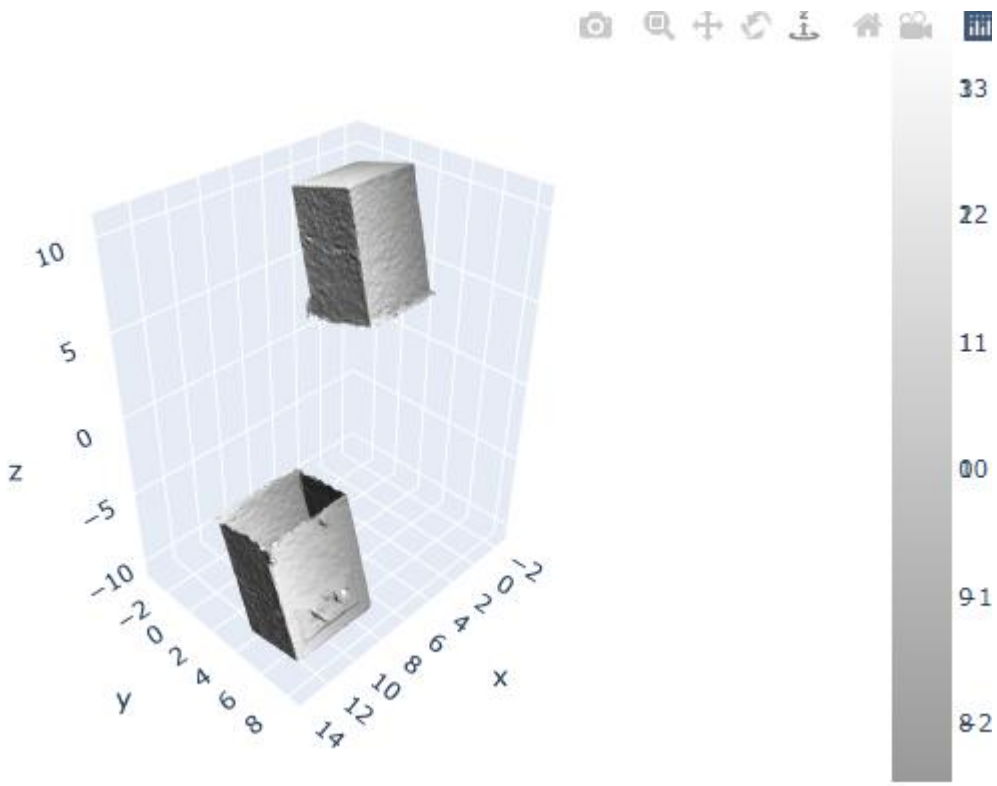
http://www.thothchildren.com/chapter/5b46226e103f2f316870c401

```
# 転換[変換]: 同次変換行列して、回転と並進移動
def transform(mesh):
    T = np.eye(4)    # 単位行列
    T[:3, :3] = o3d.geometry.get_rotation_matrix_from_xyz((np.pi, 0, 0
))   # 回転行列
    T[0, 3] = 10    # X の移動量
    T[1, 3] = 5     # Y の移動量
    print(T)   # 変換行列を表示
    mesh_t = copy.deepcopy(mesh).transform(T)    # 回転と並進移動したデータを
コピー
    o3d.visualization.draw_plotly([mesh, mesh_t])   # 元データと変換したデー
タを表示

transform(fp_mesh)   # メッシュデータを変換表示
```

同次変換行列を表示

```
[[ 1.0000000e+00  0.0000000e+00  0.0000000e+00  1.0000000e+01]
 [ 0.0000000e+00 -1.0000000e+00 -1.2246468e-16  5.0000000e+00]
 [ 0.0000000e+00  1.2246468e-16 -1.0000000e+00  0.0000000e+00]
 [ 0.0000000e+00  0.0000000e+00  0.0000000e+00  1.0000000e+00]]
```


10
5
0
−5
−10
z
−2
0
2
4
6
8
y
14
12
10
8
6
4
2
0
−2
x
33
22
11
00
91
82