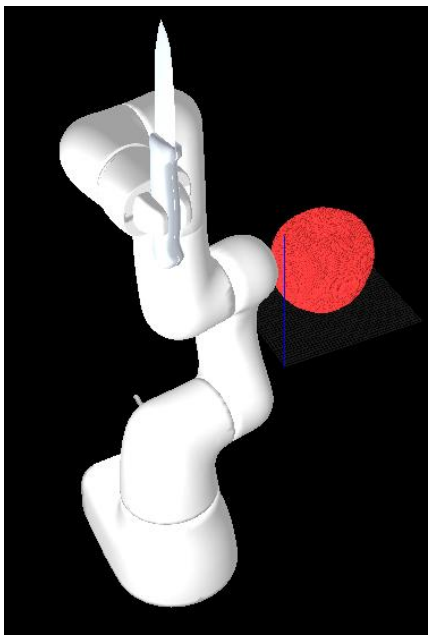
Apple Cut by Cobotta based on Weak Coupling of Mujoco and MLS-MPM in Taichi

21, March, 2026

Taichi の環境で Mujoco を利用して Cobotta を制御し，mls-mpm の弾性体粒子リンゴを切る．ナイフはマルチボディ．Cobotta とナイフは位置制御．ナイフとリンゴは双方向力学計算しているが，ナイフへの反力は単純化のためいったん無視．Cobotta とリンゴは連成（衝突計算）していない．ナイフとリンゴは Python scope（CPU ベース）で trimesh の関数を使っているため遅い．本気でやるなら，ここを Taichi scope（GPU ベース）のコードに書き換える必要がある．



(1) Mujoco のインストール

Mujoco はマルチボディ対応の物理エンジン．リンクマニピュレータの運動生成に強い．

．mujoco 入門

https://rikei-tawamure.com/entry/2025/05/24/211341

．mujoco リファレンス

https://mujoco.readthedocs.io/en/stable/XMLreference.html?utm_source=chatgpt.com

Taichi の環境に Mujoco を入れる．両方の環境を使えるようにしなければならない．Taichi が activate されている環境で，pip を update して mujoco をインストール．mujoco-py もあるようだが，サポートが終了している．間違いやすいので注意．Taichi のインストールは以下参照．

https://www.kikulab.chibatech.ac.jp/wordpress/wp-content/uploads/2026/02/Taichi-2.pdf

```
> source taichi_env/bin/activate
> python -m pip install -U pip                # pip update
> python -m pip install taichi mujoco         # taichi に mujoco をインストール．
```

(2) 実行

実行ファイル

```
CAD/****.obj            # CAD モデルは obj.　Cobotta の各関節毎にある．ナイフも．リンゴも．
     /****.stl          # CG 用に Mujoco がバイナリ stl を使う．
     /****.jpg          #  テクスチャ用．無くてもよい
engine/MPM_solver.py    # mls-mpm の力学計算メインソルバ
       /****.py
cobotta.xml             # cobotta の ml 形式で書かれた Json ファイル．関節の位置関係，自由度設定
cobottaCut01.py         #  メインコード
GUI.py                  #  ディスプレイのグラフィック用
motion.py               #  ナイフとリンゴの連成力学計算
parameters.py           #  計算条件他設定ファイル
STL.py                  #  ナイフ可視化用 stl 保存ファイル
VTU.py                  #  粒子可視化用 VTU 保存ファイル
```

これらがあるディレクトリで実行

```
> python3 cobottaCut01.py
```

VTU ディレクトリが作成され，可視化用 VTU ファイルが保存されていく．
実行中，window がオープンして GUI で描画できるようになっているが，SSH などで外から実行するときには描画なしで実行する．

(3) 可視化

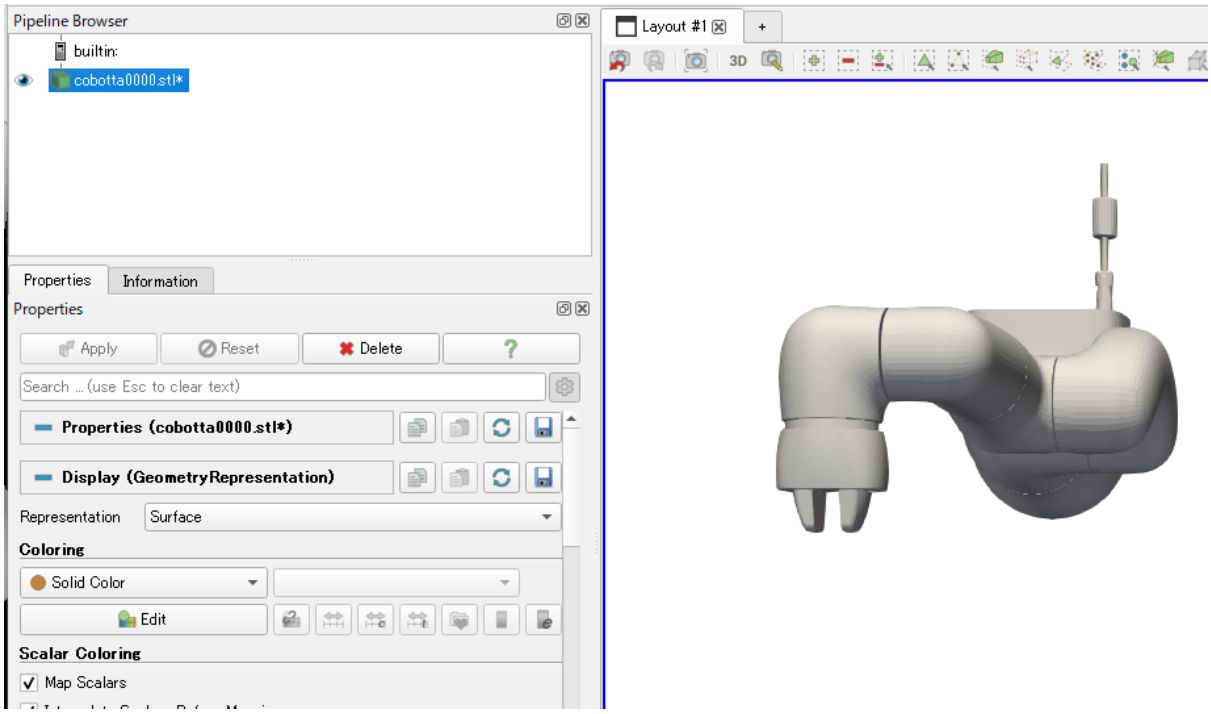
Paraview で可視化する．

. Cobotta は出力された時系列バイナリ stl ファイルをそのまま読んでデフォルト表示．テクスチャは無し．obj の時系列ファイルは読めないという謎のポリシーにより，わざわざ stl に変換して保存している．
. リンゴは，粒子なので Gaussian resampling と contour で表示．切ったもの（トポロジー，つまり，切断により境界が増えたり減ったりするもの）にテクスチャ画像が対応していない．テクスチャを貼りたければ以下を参照．
  https://www.kikulab.chibatech.ac.jp/wordpress/wp-content/uploads/2026/03/MPMlizard01.pdf
. ナイフは，obj の uv に対応した時系列 vtu ファイルを新たに python プログラムで変換，生成して表示．この場合，時系列 stl ファイルは使わない．テクスチャを使わないなら Cobotta と同じ．stl を直接読み込んで表示．変換プログラムは，以下．chatGPT さん完全作成．
  obj_stl_uv_to_vtp.py
  VTU ディレクトリ内で，knife***.stl ファイルを読み込んで，CAD/knife00.obj に対応させて作成する．

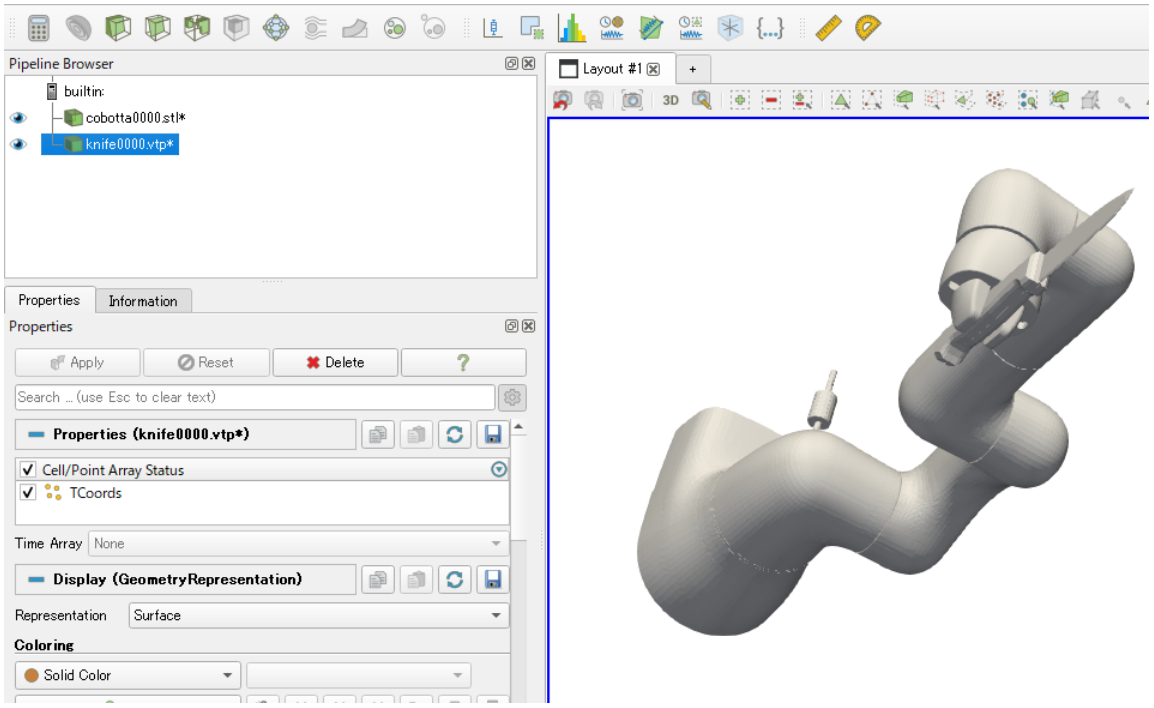
```
> python3 obj_stl_uv_to_vtp.py ¥
--obj ../CAD/knife00.obj ¥          # テンプレートとなるナイフの obj ファイルパス
--ref-stl knife0000.stl ¥           # 参照する最初のナイフの stl ファイル
--stl-glob "knife*.stl" ¥           # ナンバリングされている数だけ stl ファイルを読む
--outdir knife_vtp ¥                # 保存先のディレクトリ名
--dt 0.1                            # 時間刻み．実質使わない．
```

まとめるとこういうこと：
obj 形式：CAD データテキストファイル．テクスチャを貼れるが，paraview の時系列読み込みに未対応．Mujoco が CG 用に使う．
stl 形式:CAD データファイル．バイナリでもテキストでも OK. Mujoco は衝突計算などに使う．paraview の時系列読み込みに対応．テクスチャは貼れない．
VTU 形式：paraview 等可視化用ファイル．xml ベース．非構造格子に対応．点群処理他，可視化に強い．

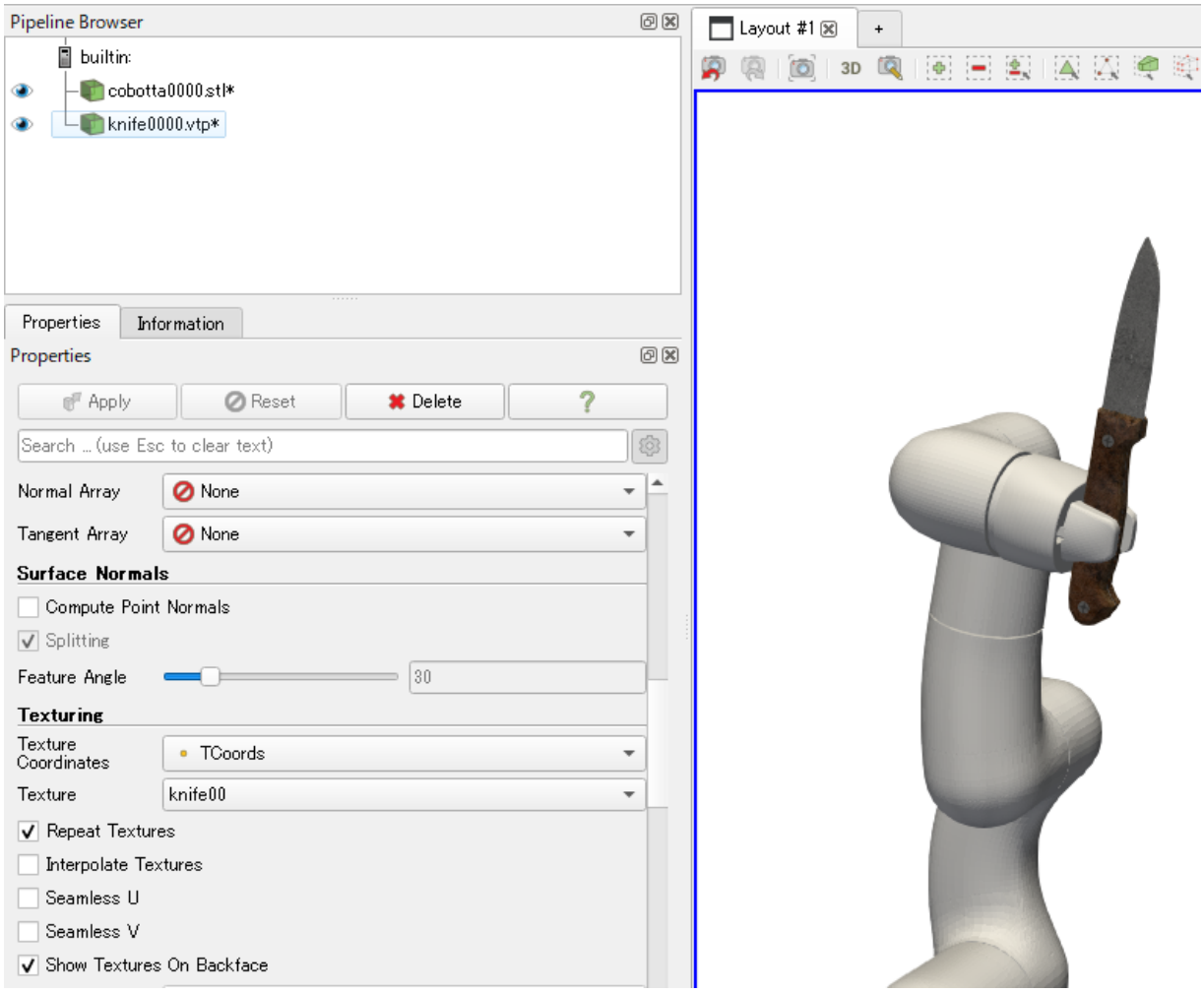
以上の準備が整ったら，以下の通り可視化する．paraview を立ち上げる．
File->Open
から Group として時系列データをまとめて読み込む．



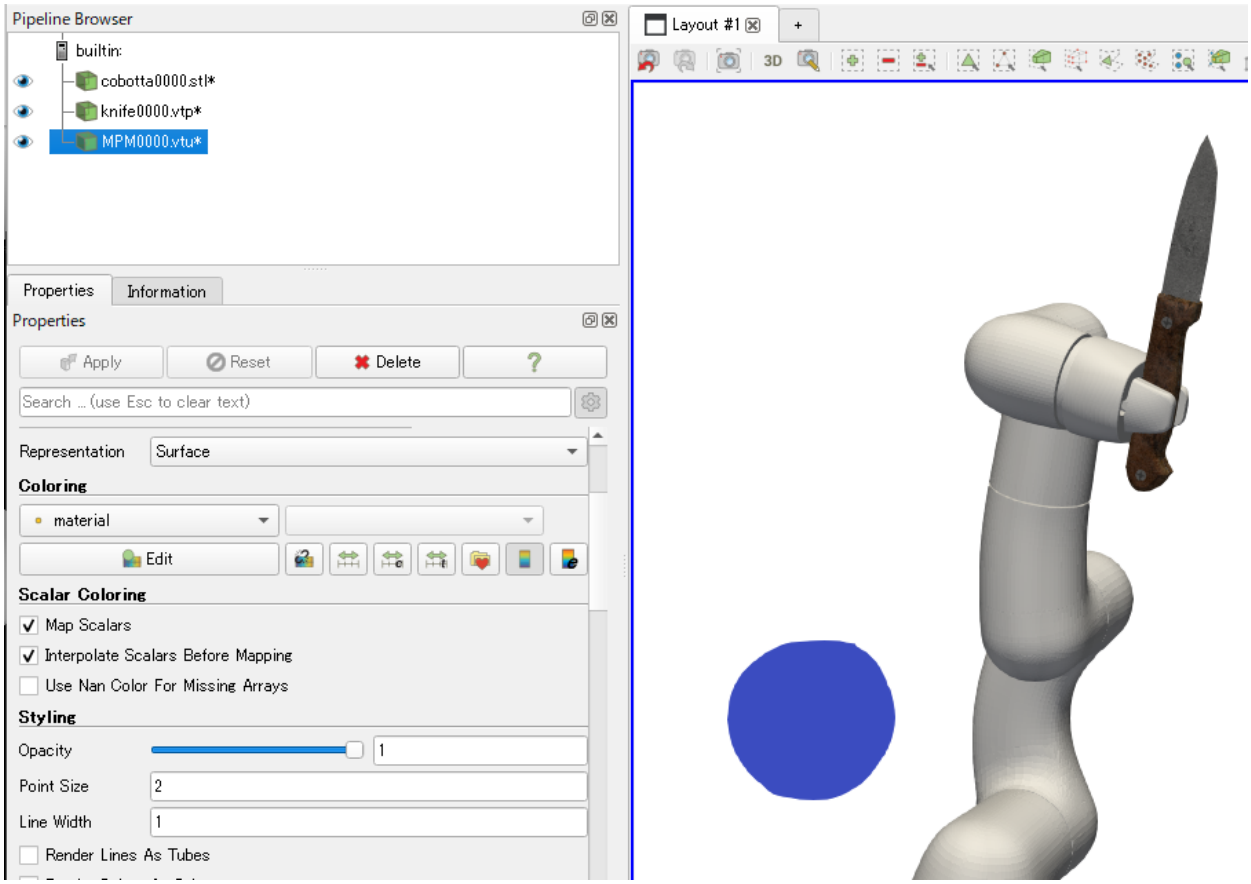
次に，ナイフの情報を読み込む．一つ下の階層の knife_VTU のディレクトリにあるので時系列で同様に vtu ファイルとして読み込む．



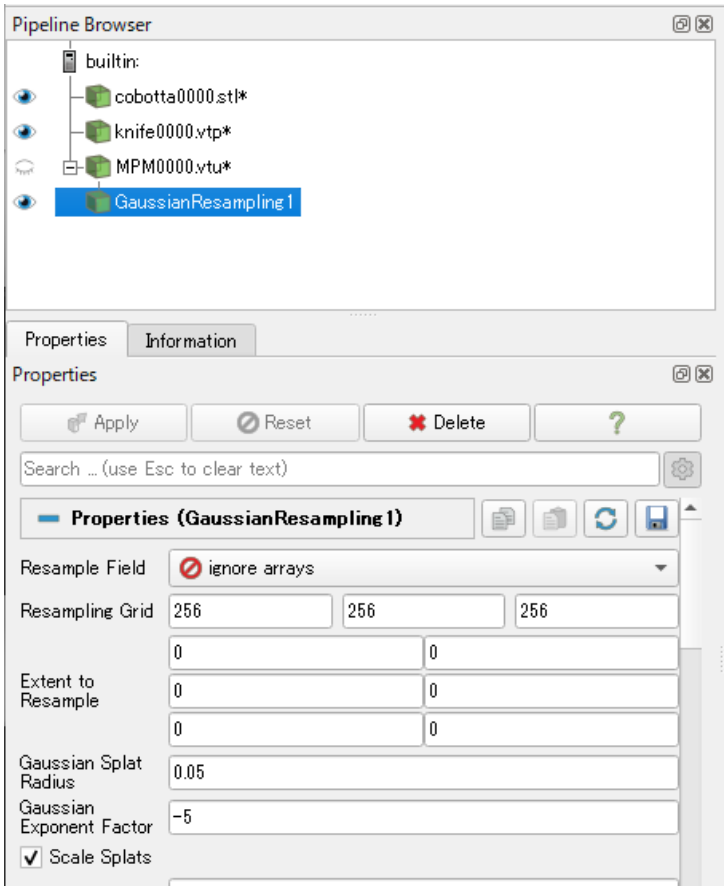
Texuring->Texture で knife00.jpg を読み込むと，ナイフが表示される．



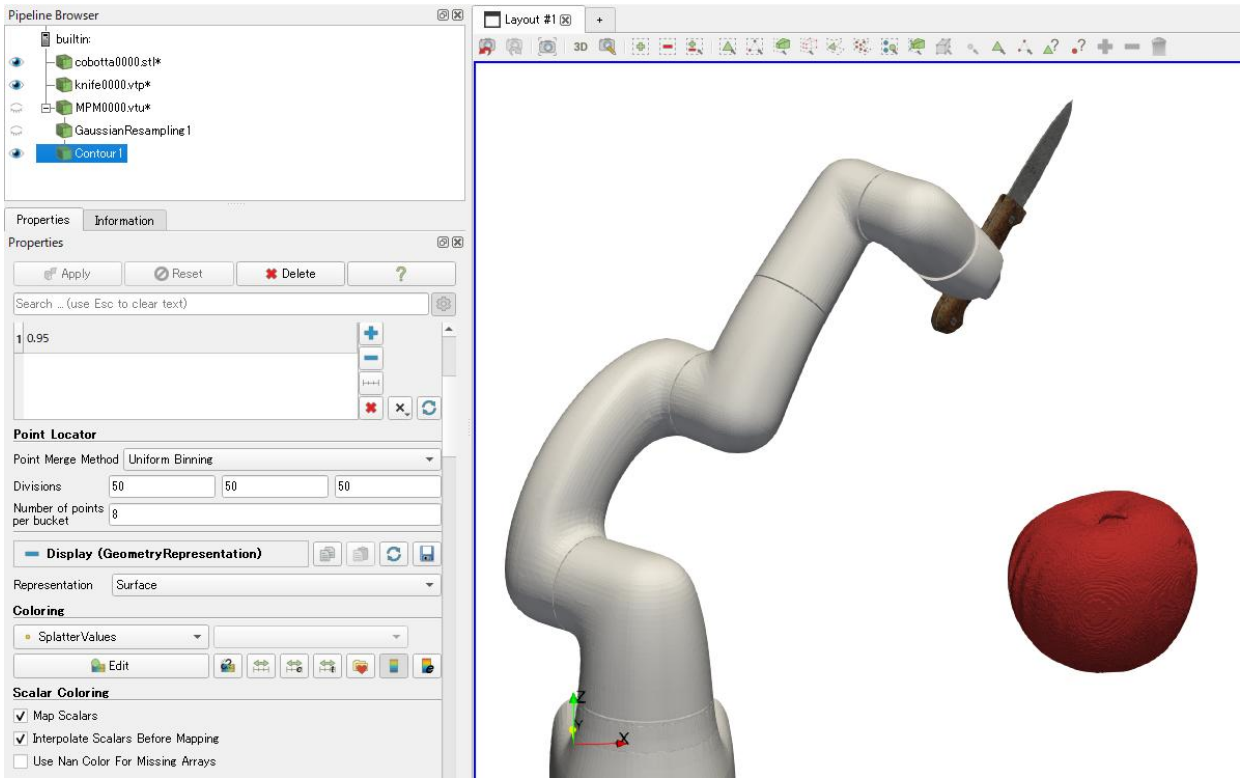
同様にリンゴ（MPM****.vtu）を時系列で読み込む.
Coloring->material
にすると，点描画で描かれる.



Filters->Alphabetical->Gaussian resampling
で解像度（Resampling Grid）を 256 に，Gaussian splat Radius を 0.05 にして Apply する.
リンゴの解像度を 128 にして計算しているので，さらに細分化して描画していることになる.



Contour で値を 0.95（粒子が 1 なのでほとんど粒子よりで描画）に設定し，Coloring で適当に設定して赤にする．



File->save animation
で名前を付け，fps を 30 にして動画を作成できる．

(4) MuJoCo による Cobotta の制御
．設定ファイル（cobotta.xml）
各リンクをマルチボディとして設定して，関節で自由度を与えて結合している．衝突などの力学計算にバイナリの stl を使い，CG 用にテキストの obj ファイルを使っている．ここでは，ナイフを持たせる座標系の原点と姿勢行列を取得するために，site（=”ee”）という点を手先に設定している．
メインコードでは，para.XML に記載のファイル名（cobotta.xml）の設定を読み込んで，model と data にする．data の qpos[i]が i 番目の関節変数．mj_step で dt 秒時間積分する．関節変数値を変更したら，時間積分するか，mj_kinematics を実行しないと，関節の位置姿勢は更新されない．

```
model = mujoco.MjModel.from_xml_path(para.XML)
data = mujoco.MjData(model)
data.qpos[i]                                    # i 関節変位
data.qval[i]                                    # i 関節速度
mujoco.mj_step(model, data)                     #  力学計算
data.site("ee").xpos.copy()                     # ee の位置ベクトル
data.site("ee").xmat.reshape(3, 3).copy()       # ee の姿勢行列
model.body_pos[i]                               # i 番目のボディの位置
model.body_quat[i]                              # i 番目のボディの姿勢
```

．cobotta.xml

```
<mujoco model="cobotta">                                        #  モデル名
  <compiler angle="radian" meshdir="./" autolimits="true"/>     #  角度は radian

  <option integrator="implicitfast"/>                           #  陰的積分

  <default>
    <default class="cobotta">                                   #  クラス名
      <material specular="0.5" shininess="0.25"/>               #  デフォルト関節パラメータ設定
      <joint armature="0.1" damping="1" axis="0 0 1" range="-3.141592 3.141592"/>
      <general  dyntype="none"  biastype="affine"  ctrlrange="-3.141592  3.141592"  forcerange="-87
87"/>
      <default class="finger">                                  #  指の稼働軸と範囲設定
        <joint axis="0 1 0" type="slide" range="0 0.04"/>
      </default>

      <default class="visual">                                  #  可視化設定
        <geom type="mesh" contype="0" conaffinity="0" group="2"/>
      </default>
      <default class="collision">                               #  衝突設定，位置制御だと意味なし．
        <geom type="mesh" group="3"/>
        <default class="finger_collision_1">
          <geom type="box" size="0.0085 0.004 0.0085" pos="0 0.0055 0.0445"/>
        </default>
        <default class="finger_collision_2">
```

```
        <geom type="box" size="0.003 0.002 0.003" pos="0.0055 0.002 0.05"/>
      </default>
    </default>
  </default>
</default>

<asset>
  <material class="cobotta" name="white" rgba="1 1 1 1"/>            # 色定義
  <material class="cobotta" name="off_white" rgba="0.901961 0.921569 0.929412 1"/>
  <material class="cobotta" name="black" rgba="0.25 0.25 0.25 1"/>

  <!-- Collision meshes -->                                          # 衝突計算用バイナリ stl
  <mesh name="base00_c" file="CAD/base00.stl"/>
  <mesh name="link01_c" file="CAD/link01.stl"/>
  <mesh name="link02_c" file="CAD/link02.stl"/>
  <mesh name="link03_c" file="CAD/link03.stl"/>
  <mesh name="link04_c" file="CAD/link04.stl"/>
  <mesh name="link05_c" file="CAD/link05.stl"/>
  <mesh name="hand_c" file="CAD/hand.stl"/>
  <mesh name="finger00_c" file="CAD/finger00.stl"/>
  <mesh name="finger01_c" file="CAD/finger01.stl"/>

  <!-- Visual meshes -->                                             # 可視化用テキスト obj ファイル
  <mesh file="CAD/base00.obj"/>
  <mesh file="CAD/link01.obj"/>
  <mesh file="CAD/link02.obj"/>
  <mesh file="CAD/link03.obj"/>
  <mesh file="CAD/link04.obj"/>
  <mesh file="CAD/link05.obj"/>
  <mesh file="CAD/hand.obj"/>
  <mesh file="CAD/finger00.obj"/>
  <mesh file="CAD/finger01.obj"/>
</asset>

<worldbody>
  <light name="top" pos="0 0 2" mode="trackcom"/>                    # 照明位置
  #base00 が土台の名前．pos が設置位置，euler がオイラー角で姿勢設定，
  <body name="base00" pos="-0.24 -0.22 0" euler="1.570796 0 0" childclass="cobotta">
    mass が質量，pos が重心，fillinertia が完成テンソル
    <inertial mass="0.2" pos="0.001 0.045 -0.022" fullinertia="0.02 0.02 0.02 0.0 0.0 0.0"/>
    <geom mesh="base00" material="white" class="visual"/>            # 色
    <geom mesh="base00_c" class="collision"/>                        # 衝突クラスとモデル洗濯

    #これ以降，ぶら下がっているのは，親の座標系に従った位置姿勢
    <body name="link01" pos="0 0 0">
      <inertial mass="0.1" pos="0.007 0.158 0.0" fullinertia="0.01 0.01 0.01 0.0 0.0 0.0"/>
      <joint name="joint1" type="hinge" pos="0 0 0" axis="0 1 0" limited="true"/>
      <geom mesh="link01" material="off_white" class="visual"/>
      <geom mesh="link01_c" class="collision"/>

      <body name="link02" pos="0 0.18 0">
        <inertial mass="0.1" pos="0.091 0.256 -0.013" fullinertia="0.01 0.01 0.01 0.0 0.0 0.0"/>
        <joint name="joint2" type="hinge" pos="0 0 0" axis="1 0 0" limited="true"/>
        <geom mesh="link02" material="off_white" class="visual"/>
        <geom mesh="link02_c" class="collision"/>

        <body name="link03" pos="0 0.165 0">
          <inertial mass="0.1" pos="0.026 0.202 -0.005" fullinertia="0.01 0.01 0.01 0.0 0.0 0.0"/>
          <joint name="joint3" type="hinge" pos="0 0 0" axis="1 0 0" limited="true"/>
          <geom mesh="link03" material="white" class="visual"/>
```

```
            <geom mesh="link03_c" class="collision"/>

            <body name="link04" pos="0 0 0">
              <inertial mass="0.1" pos="0.019 0142 -0.012" fullinertia="0.01 0.01 0.01 0 0 0"/>
              <joint name="joint4" type="hinge" pos="0.02 0 -0.015" axis="0 1 0" limited="true"/>
        #"0.02 0.357 0.083" CAD の値と合わない.
              <geom mesh="link04" material="off_white" class="visual"/>
              <geom mesh="link04_c" class="collision"/>

              <body name="link05" pos="-0.045 0.178 -0.012">        # 0.045 0.178 0.012
                <inertial mass="0.1" pos="0.004 0.0 0.008" fullinertia="0.01 0.01 0.01 0.0 0.0 0.0"/>
                <joint name="joint5" type="hinge" pos="0 0 0" axis="1 0 0" limited="true"/>
                <geom mesh="link05" material="off_white" class="visual"/>
                <geom mesh="link05_c" class="collision"/>

                <body name="hand" pos="0 0 0.04">
                    <inertial mass="0.05" pos="0.0 0.0 0.024" fullinertia="0.005 0.005 0.005 0.0 0.0
0.0"/>
                    <joint name="joint6" type="hinge" pos="0 0 0" axis="0 0 1" limited="true"/>
                    <geom mesh="hand" material="off_white" class="visual"/>
                    <geom mesh="hand_c" class="collision"/>

                    <body name="finger00" pos="0.01 0.0 0.031" euler="1.570796 0 0">          #
0.018 0.0 0.042
                        <inertial mass="0.05" pos="0.003 0.015 0" fullinertia="0.005 0.005 0.005 0.0
0.0 0.0"/>
                        <joint name="joint7" type="slide" pos="0 0 0" axis="1 0 0" limited="true"/>
                        <geom mesh="finger00" material="off_white" class="visual"/>
                        <geom mesh="finger00_c" class="collision"/>
                    </body>
                    <body name="finger01" pos="-0.01 0 0.031" euler="1.570796 3.141592 0">
                        <inertial mass="0.05" pos="0.003 0.015 0" fullinertia="0.005 0.005 0.005 0.0
0.0 0.0"/>
                        <joint name="joint8" type="slide" pos="0 0 0" axis="1 0 0" limited="true"/>
                        <geom mesh="finger01" material="off_white" class="visual"/>
                        <geom mesh="finger01_c" class="collision"/>
                    </body>
                    <site name="ee" pos="0 0 0.05" quat="1 0 0 0"/> ナイフを持つ位置姿勢用
                </body>
              </body>
            </body>
          </body>
        </body>
      </body>
    </body>
  </worldbody>

  <actuator>      # 関節アクチュエータの設定.
    <general class="cobotta" name="actuator1" joint="joint1" gainprm="4500" biasprm="0 -4500 -
450"/>
    <general class="cobotta" name="actuator2" joint="joint2" gainprm="4500" biasprm="0 -4500 -
450"/>
    <general class="cobotta" name="actuator3" joint="joint3" gainprm="3500" biasprm="0 -3500 -
350"/>
    <general class="cobotta" name="actuator4" joint="joint4" gainprm="3500" biasprm="0 -3500 -
350"/>
    <general class="cobotta" name="actuator5" joint="joint5" gainprm="2000" biasprm="0 -2000 -
200"/>
    <general class="cobotta" name="actuator6" joint="joint6" gainprm="2000" biasprm="0 -2000 -
200"/>
```

```
    <general class="cobotta" name="actuator7" joint="joint7" gainprm="2000" biasprm="0 -2000 -200"/>
    <general class="cobotta" name="actuator8" joint="joint8" gainprm="2000" biasprm="0 -2000 -200"/>
  </actuator>

  <keyframe>    # ホーム位置姿勢設定
    <key name="home" qpos="0 0 0 0 0 0 0 0" ctrl="0 0 0 0 0 0 0 0"/>
  </keyframe>
</mujoco>
```

. シミュレーション条件設定ファイル（parameters.py）

```
import taichi as ti

# ***** Parameters **********
PI = ti.math.pi                 # 円周率の定義．Taichi scope
SimulationTime = 2.0            # [s] シミュレーション時間
dt = 2e-4                       # [s] dt, 時間積分刻み
CalculationSpace = 0.2          # [m]    粒子部分の計算空間
Resolution = 128                # 計算空間の解像度，2^n 刻み．高解像度は 256，低解像度は 64
gravity = ( 0.0, 0.0, -9.81 )   # [m/s^2] 重力
myu = 0.3                       # 摩擦係数，ナイフ＆リンゴ間と，リンゴ＆床間
epsilon = 0.1                   # restitution coefficient，跳ね返り係数
airresistance = 1.0             # ナイフの空気抵抗，空間に空気はない．
VTUinterval = 50                # VTU saving interval データ保存インターバル

XML = "cobotta.xml"             # Mujoco format file

apple = {                       # リンゴの設定
        "fileName": 'CAD/apple.obj',
        "translation": [0.1, 0.1, 0.072],       # 平行移動
        "scale": [1, 1, 1],                     # スケール
        "mass": 0.2,                            # [kg] 質量
        }

knife = {                       # ナイフの設定
        "fileName": 'CAD/knife00.obj',
        "translation": [0.0,0.0,0.05],
        "rotation": [PI/2.0, 0, 0.0],           # Euler angle [rad]
        "scale": [1, 1, 1],
        "COGx": [0, 0, 0],                      # 初期重心位置
        "COGv": [0, 0, 0],                      # 初期重心速度
        "mass": 0.2,
        "inertia": [[0.00001,0,-0.00001],[0,0.0001,0],[-0.00001,0,0.0001]],     # 慣性テンソル
        }
```

. メインコード（cobottaCut01.py）

```
import os, sys                                  # インポート関係
import mujoco
import taichi as ti
import numpy as np                              # python scope 配列関係
import xml.etree.ElementTree as ET
import trimesh                                  # obj, stl, 粒子関係読み込み書き出し．

import glob
from collections import defaultdict
from engine.mpm_solver import MPMSolver
```

```
from VTU import VTUclass
from motion import Motionclass
from trimesh.exchange.export import export_mesh
from STL import export_mujoco_to_single_stl

# Parameters
import parameters as para                                    #  設定パラメータ読み込み

ti.init(arch=ti.cpu)                                         # CPU 計算
import GUI                                                   #  画像出力読み込み

# *** for MuJoCo ***    Cobotta 読み込み
if not os.path.exists(para.XML):
    print("ERROR: cobotta.xml not found in current directory.")
    sys.exit(1)
BASE = os.path.dirname(os.path.abspath(para.XML))

# parse XML for asset meshes
tree = ET.parse(para.XML)
root = tree.getroot()
asset = root.find("asset")
mesh_name_to_file = {}
if asset is not None:
    for m in asset.findall("mesh"):
        name = m.get("name")                # may be None
        file = m.get("file")
        if file:
            mesh_name_to_file[name] = os.path.join(BASE, file)
# fallback: any file= occurrences
if not mesh_name_to_file:
    for elem in root.iter():
        if 'file' in elem.attrib:
            fname = elem.attrib['file']
            mesh_name_to_file[os.path.splitext(os.path.basename(fname))[0]] = os.path.join(BASE,
fname)

print("Meshes (from XML):")
for k,v in mesh_name_to_file.items():
    print(" ", repr(k), "->", v)

# load meshes (skip materials to avoid Pillow dependency)
meshes = {}
for name, path in mesh_name_to_file.items():
    if not os.path.exists(path):
        print(" WARNING: mesh file not found:", path)
        continue
    try:
        tm = trimesh.load(path, force='mesh', skip_materials=True)
        if isinstance(tm, trimesh.Scene):
            tm = trimesh.util.concatenate(tm.dump())
        tm.vertices = tm.vertices.astype(np.float32)
        tm.faces = tm.faces.astype(np.int32)
        meshes[name] = tm
        print(f" Loaded '{name}' -> verts {tm.vertices.shape}, faces {tm.faces.shape}")
    except Exception as e:
        print(" Failed to load", path, ":", e)

if len(meshes) == 0:
    print("No meshes loaded. Ensure mesh files are present next to cobotta.xml.")
```

```
# build geom->(mesh_name, body_name, pos, quat) mapping
geom_map = GUI.set_geom_map(root)

# init MuJoCo
model = mujoco.MjModel.from_xml_path(para.XML)
data = mujoco.MjData(model)
body_name_to_id = {model.body(i).name:i for i in range(model.nbody)}
print("Bodies (index:name):")
for i in range(model.nbody):
    print(" ", i, ":", model.body(i).name)

# *** for mls-mpm ***  リンゴの設定
mpm = MPMSolver(res=(para.Resolution, para.Resolution, para.Resolution),
size=para.CalculationSpace, max_num_particles=2 ** 22, use_ggui=True)
mpm.add_CADmodel(para.apple["fileName"], offset=para.apple["translation"],
material=mpm.material_elastic)

mpm.set_gravity(para.gravity)

# *** multi-body ***  ナイフの設定
mujoco.mj_kinematics(model, data)
p_ee0 = data.site("ee").xpos.copy()                 # end effector position
R_ee0 = data.site("ee").xmat.reshape(3, 3).copy()   # end effector posture
print("p_ee0 ", p_ee0 )
print("R_ee0 ", R_ee0 )

knifemotion = Motionclass(para.knife, mpm, para.dt, p_ee0, R_ee0 )
faces = np.asarray(knifemotion.mesh.faces.astype(np.int32), dtype=np.int64)
print("COGx ", knifemotion.COGx[None] )
print("R ", knifemotion.R[None] )

mesh_buffers = GUI.set_mesh_buffers(meshes)

#  シミュレーション
t = 0
cnt = 0
cnt2 = 0

#os.makedirs("image", exist_ok=True)
os.makedirs("VTU", exist_ok=True)
vtu = VTUclass()

while t < para.SimulationTime:
    print(f"time={t:.4f}")

    # joint motion    Cobotta の関節変数制御
    if t < 1.0:
        data.qpos[0] = 135.0/180.0*para.PI * t
        data.qpos[1] =   45.0/180.0*para.PI * t
        data.qpos[2] = 0.0
    else:
        data.qpos[0] = 135.0/180.0*para.PI
        data.qpos[1] =   45.0/180.0*para.PI + 60.0/180.0*para.PI * (t-1.0)
        data.qpos[2] = -10.0/180.0*para.PI * (t-1.0)
    data.qpos[3] = 0.0
    data.qpos[4] = 0.0
    data.qpos[5] = 0.0
    data.qpos[6] = 0.0
    data.qpos[7] = 0.0
```

```
        mujoco.mj_step(model, data)                # cobotta 運動学計算
        p_ee = data.site("ee").xpos.copy()
        R_ee = data.site("ee").xmat.reshape(3, 3).copy()
        dR_ee = R_ee @ R_ee0.transpose()
        dx = p_ee - p_ee0

        dx_ti = ti.Vector(np.asarray(dx, dtype=np.float32).tolist())
        dR_ti = ti.Matrix(np.asarray(dR_ee, dtype=np.float32).tolist())
        knifemotion.update( dx_ti, dR_ti, p_ee0 ) # cobotta の運動に伴うナイフの位置姿勢更新

        p_ee0 = p_ee
        R_ee0 = R_ee

        nearestx, inside_pid = knifemotion.collision( mpm.x )
        mpm.step(para.dt, knifemotion, nearestx, inside_pid)

        if cnt%para.VTUinterval == 0:    データ保存ルーチン
            GUI.render(mpm, geom_map, mesh_buffers, body_name_to_id, data, knifemotion.obj_vertices,
knifemotion.indices )
            GUI.window.show()

            vtu.write_vtuMPMASCII(f"./VTU/MPM{cnt2:04d}.vtu", mpm )
            verts = np.asarray(knifemotion.vertices.to_numpy(), dtype=np.float64)
            mesh = trimesh.Trimesh(vertices=verts, faces=faces, process=False, validate=False, )
            export_mesh( mesh, f"./VTU/knife{cnt2:04d}.stl", file_type="stl", )                    #
"stl_ascii"

            export_mujoco_to_single_stl( model, data, f"./VTU/cobotta{cnt2:04d}.stl",
                    infinite_plane_half_extent=2.0, )   # floor が infinite plane なら適当に有限化
            # save frame
            # GUI.window.save_image(f"./image/frame{cnt2:04d}.png")

            cnt2 += 1

        t += para.dt
        cnt += 1
```

. ナイフとリンゴの衝突計算（motion.py）

双方向の連成計算をしているが, ナイフは位置制御される Cobotta と同期しているので実質一方向計算. リンゴは 20cm の空間を 128 分割した格子上にあり，格子は格子方向に 2 個の粒子で構成されるので, 粒子間隔は，200mm/128/3=0.52mm となる．このナイフの刃の幅が 0.2mm なので基本すり抜ける．刃幅は 1mm 以上ほしい．詳細は以下を参照.

https://www.kikulab.chibatech.ac.jp/wordpress/wp-content/uploads/2026/03/MPMapple.pdf