UPDATED: 12/30/20 -- compatible with robosuite version 1.1.0

For guidance on how to import a novel gripper model, see THIS tutorial

Assumptions: (Make sure these are all met!)

- Raw model is in .xacro or .urdf form (e.g.: from ROS)
- Corresponding element meshes exist somewhere

Steps

1. Convert .xacro to .urdf file via ROS
    a. "rosrun xacro xacro PATH_TO_XACRO.xacro > PATH_TO_TARGET.urdf"

2. Convert .urdf to .xml Mujoco file
    a. If the mesh files are not in .STL format, convert them into .STL (use a mesh converter, like this one)
    b. Open .urdf file and add the following under the <robot> tag:

```
<mujoco>
    <compiler
    meshdir="PATH_TO_MESHES_DIRECTORY"
    balanceinertia="true"
    discardvisual="false" />
</mujoco>
```

    c. If the meshes references were originally to the .DAE (or some other non .STL) file, change the mesh filenames to .STL in the .urdf file
    d. In a terminal, change directory to your Mujoco installation / bin directory, and execute the following:

        ■ ./compile PATH_TO_URDF.urdf PATH_TO_OUTPUT_DIR/robot.xml

3. Import the robot and mesh into the working robosuite directory
    a. Create a new directory under robosuite/models/assets/robots, ideally named after some reference to the robot arm you're importing into robosuite.
    b. Move your meshes directory as well as your newly created mujoco robot.xml file into this directory

4. Clean up the xml (this is where the hacky stuff begin :D)
    a. When mujoco converts into its native structure xml format, it automatically includes compiler info (the <compiler> and <size> tags at the top of the xml). We don't need these since they're already specified elsewhere in robosuite. So delete them
    b. Robosuite only supports torque-based actuators (in mujoco xml-speak, these are <actuator> <motor> tags). So, if your xml is missing actuators, you'll have to manually include them yourself. For reference, you can check out the other robot.xml files for the other robots currently in robosuite to see examples of actuator structuring.
    c. Under the <asset> tag, all <mesh> tags should have their "file=" updated. Prepend "meshes/" to the beginning of all of these filenames.
    d. We want to add a dummy body at the end of the arm so we can attach some visualization sites as well as create a uniform reference for grippers to attach to. At the most nested level of the

xml (probably right after joint7 or so if you're importing a 7DOF arm), REMOVE any further nested bodies belonging to the robot's gripper (if it exists) and insert the following:

```
<body name="right_hand" pos="X Y Z" quat="QW QX QY QZ">
    <!-- This sites were added for visualization. They are all standardized between models-->
    <!-- Position mimics the gripper attachment point (right_hand) -->
    <!--  Y-axis should be perpendicular to grasping motion, and Z-axis should point out of the robot eef -->
    <site name="ee" pos="0 0 0" size="0.01 0.01 0.01" rgba="0 0 1 1" type="sphere" group="1"/>
    <site name="ee_x" pos="0 0 0" size="0.005 .1"  quat="0.707105 0.707108 0 0 " rgba="1 0 0 0" 
type="cylinder" group="1"/>
    <site name="ee_z" pos="0 0 0" size="0.005 .1" quat="0.707105 0 0 0.707108" rgba="0 0 1 0" type="cylinder" 
group="1"/>
    <site name="ee_y" pos="0 0 0" size="0.005 .1" quat="0.707105 0 0.707108 0 " rgba="0 1 0 0" 
type="cylinder" group="1"/>
    <!-- This camera points out from the eef. -->
    <camera mode="fixed" name="eye_in_hand" pos="0.05 0 0" quat="0 0.707108 0.707108 0" fovy="75"/>
    <!-- to add gripper -->
```

Note 1: that the frame of reference at the point should have the z direction facing downward (i.e.: out of the end of the hand); if the convention is flipped then please flip it back by adding the appropriate rotation with "quat= …" in the <body> tag
Note 2: You must tune the `pos` and `quat` tags (i.e.: replace X Y Z QW QX QY QZ) such that any gripper attached to your robot is aligned perfectly at the end of your robot's arm. Usually this means tuning the Z pos value. You can always test the alignment by deploying your robot in an environment and iteratively tuning it visually.
Note 3: You may have to tune the eye_in_hand position for your robot model so that its FOV roughly matches that of the other grippers

e. Add the following body tag right below the <worldbody> tag as its child, (with corresponding closing tag towards the bottom of the xml and all tags between these two tabbed inwards appropriately as children of this newly created body):

```
<body name="base" pos="0 0 0">
```

f. All robots should be supported with a robotview camera, so we need to include that here in the xml as well. Below the body tag you just created, add the following camera as the body's child:

```
<!-- robot view -->
<camera mode="fixed" name="robotview" pos="1.0 0 0.4" quat="0.653 0.271 0.271 0.653"/>
<inertial diaginertia="0 0 0" mass="0" pos="0 0 0"/>
<!-- mount attached here -->
```

Note: You may have to tune the robotview camera if there are any obstructions in the robot's form factor blocking the camera

g. Verify that all elements (with the exception of inertial elements) have a `name`. If any of them don't, add a `name` attribute for that element!

5. Create a corresponding python class for your robot
    a. Create `your_robot_name.py` in robosuite/models/robots/manipulators. This is a very simple boilerplate class with a few hardcoded presets specified, and can directly be modeled/templated from another robot file in that folder, e.g.: panda_robot.py or sawyer_robot.py
    b. Add your robot class to the imports in `__init__.py` in that folder