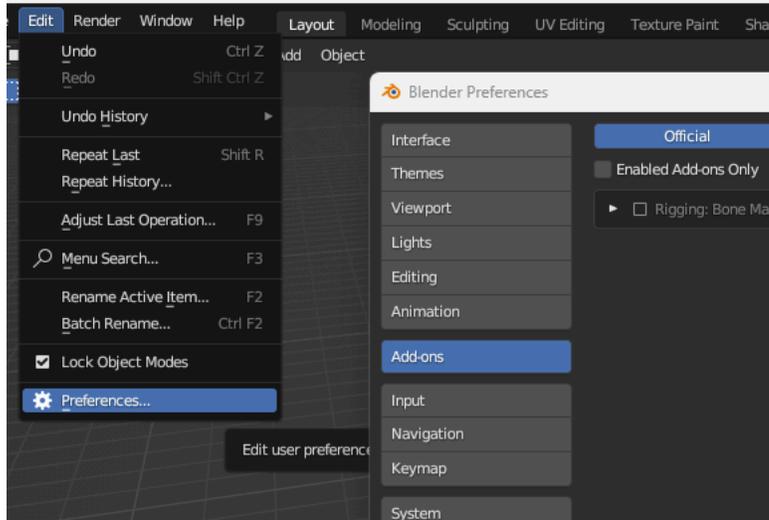
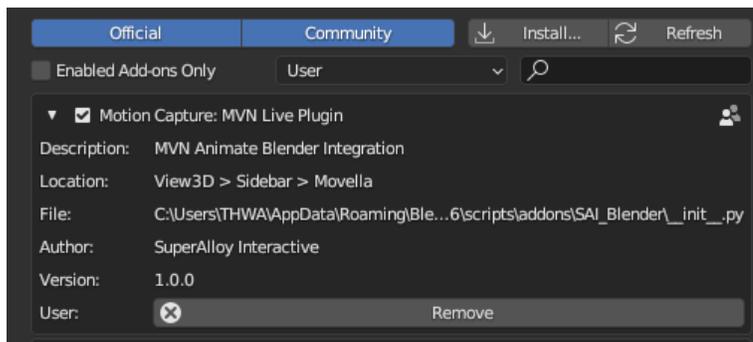


Installation

1. Open Blender. Navigate to **Edit > Preferences > Add-ons**.



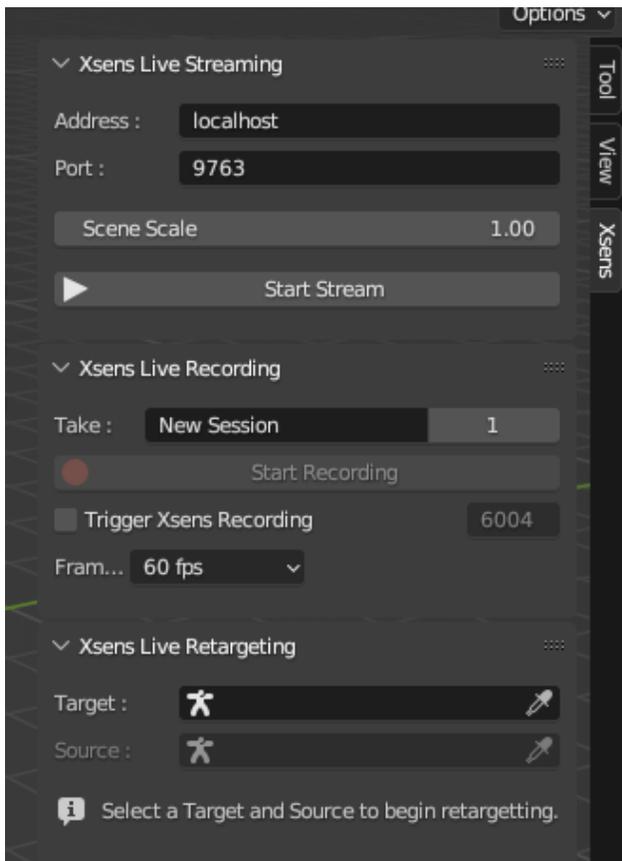
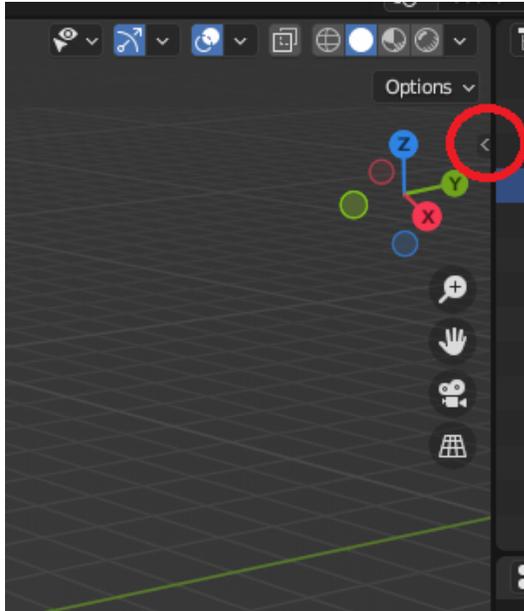
2. Click **Install...** in the top right and select the .zip of the add-on files. This will add the contents to default location: C:\Users\[USERNAME]\AppData\Roaming\Blender Foundation\Blender\3.x\scripts\addons.
3. Check the box next to “Motion Capture: MVN Live Plugin” to enable the add-on. The add-on will now be enabled for this version of Blender and can be disabled or removed at any time from this menu.



- NOTE: For this default workflow, Blender add-ons are local to the user and specific to the Blender version. If using a different user, PC, or Blender version (i.e. 3.6 -> 3.3 or 4.0), the add-on will need to be installed again.
- More info:
<https://docs.blender.org/manual/en/3.6/editors/preferences/addons.html>
- NOTE: Before removing the add-on, be sure to stop any active recording or streaming in progress to prevent issues.

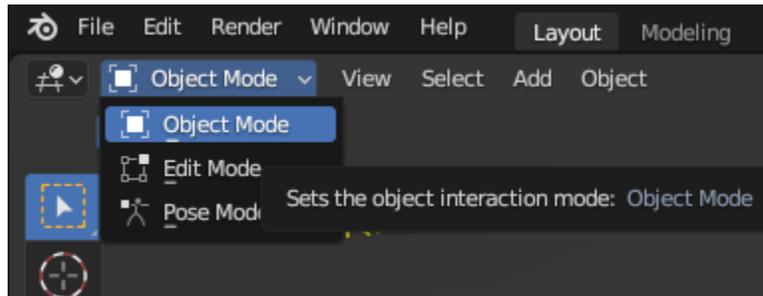
UI

The UI for the add-on is accessed through the View3D Sidebar under a new tab called “Xsens”. This can be revealed by clicking the arrow in the top right of the viewport window or by using the Blender default hotkey “N”.

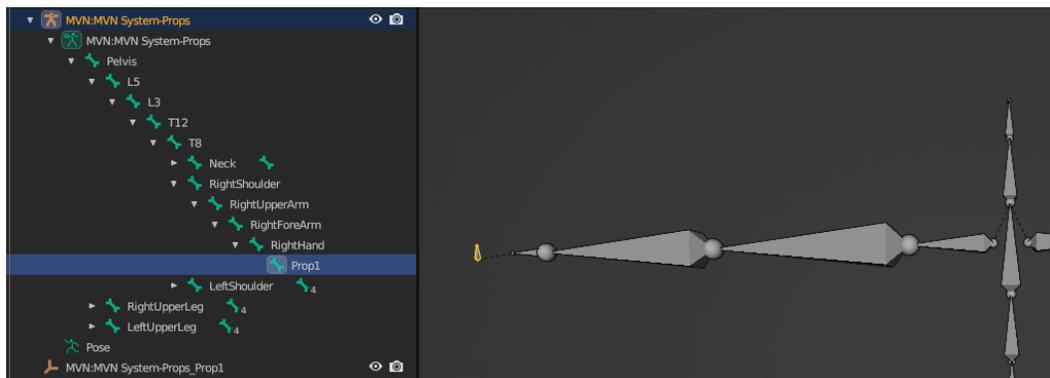


Streaming

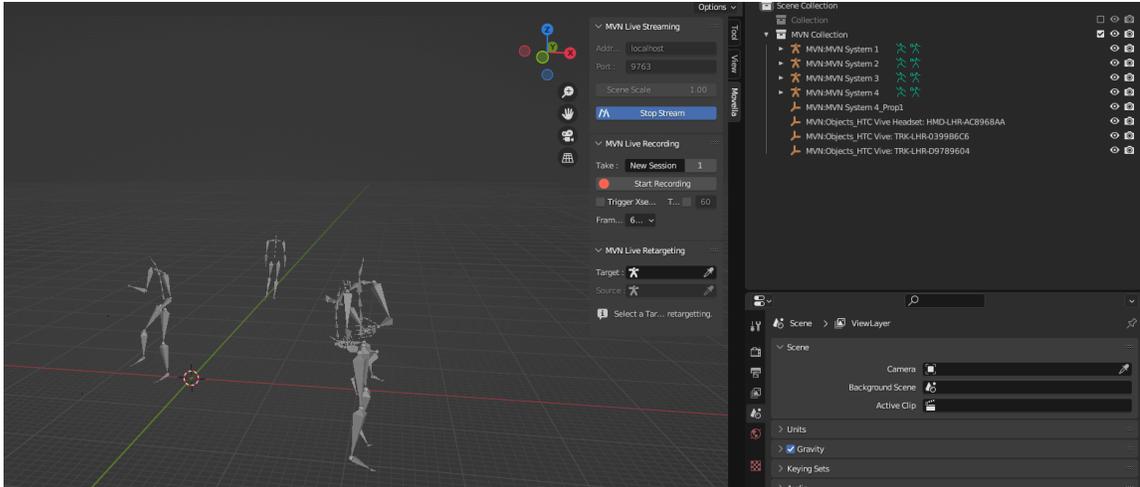
1. Set the “Address” and “Port” settings as necessary.
 - a. If MVN is on the same system, either “localhost” or “127.0.0.1” can be used for the IP address.
2. Click **Start Stream**.
 - a. Each MVN actor will be created as its own armature named “MVN:[actor name]”. If the actor has finger data, finger bones will be included in the respective armature.
 - i. NOTE: The **Start/Stop Stream** options can only be selected if the viewport is set to “Object Mode”.



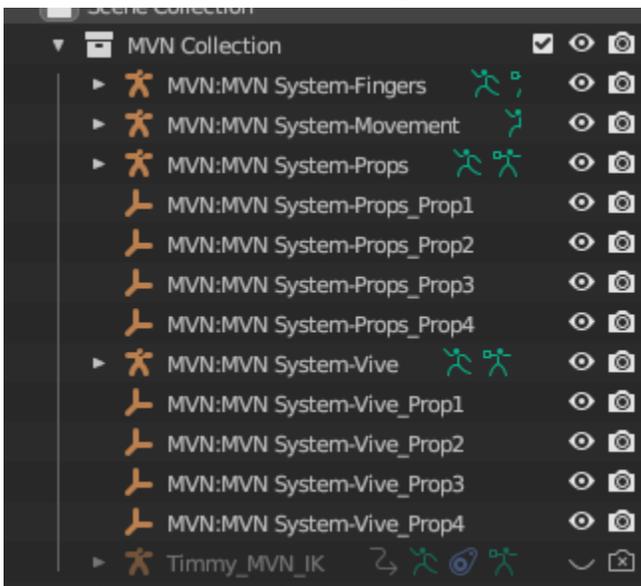
- b. Props will be created as empties named “MVN:[actor name]_Prop[#]”. They will also be added to the respective MVN armature’s hierarchy as bones named “Prop[#]” and will be parented within the hierarchy depending on the configuration in MVN.
 - i. NOTE: The bones added for props will always be created at a consistent orientation and roll (pointing up similar to the spine) regardless of where the prop is attached to in MVN.



- c. Vive objects will be created as empties named “MVN:Objects_HTC Vive [object type]: [object name]”.



- The object name will be whatever is displayed in MVN's viewport.
3. "Scene Scale" allows users to uniformly scale the MVN armatures with respect to Blender global space in order to match the scale of their own armatures.
 - a. NOTE: At the moment, only armatures are affected by "Scene Scale". Empties are not affected.
 4. All MVN Objects created through the add-on will be added to a new "MVN Collection".



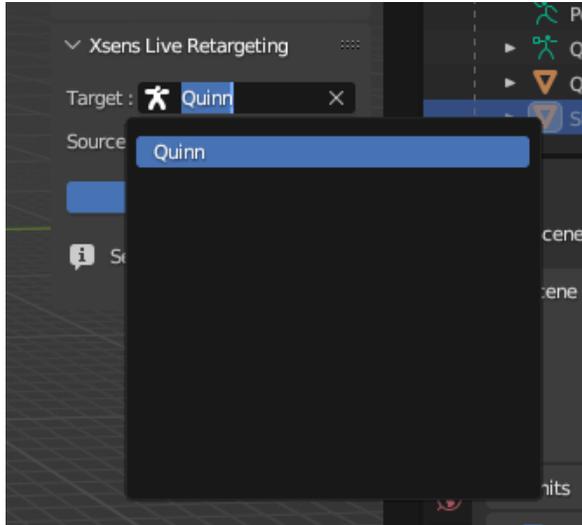
5. NOTES:
 - a. Any time the streamer is started (including stopped and restarted), all MVN armatures will be recreated. This means any references to the armatures will need to be restored (user constraints/parenting, "Source:" referencing in the retargeter, etc.). All settings to reference names are still maintained in Blender, so if the reference is reset to an object with the same name, everything should work as before (i.e. constraint settings will stay the same, bone remapping/IK settings will stay configured).
 - b. Any changes made to the MVN objects (changing, renaming, or deleting object/bones, etc.) will cause that object to stop receiving stream data unless the

changes are reverted. Objects can also be easily recreated by toggling the stream.

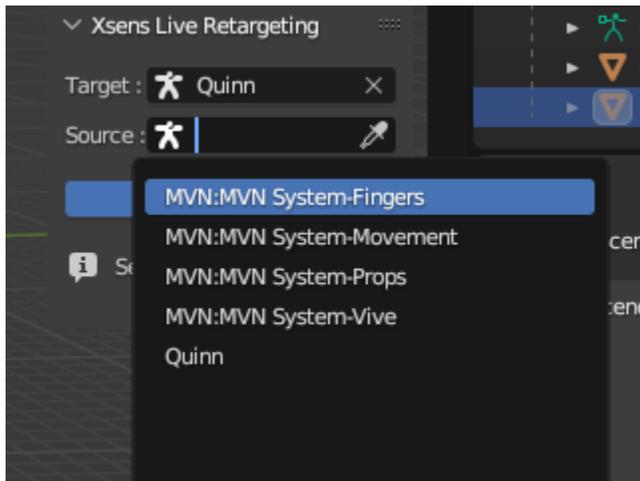
- c. If the name of the an object that would be created by the add-on already exists in the scene, the existing object will be deleted and replaced by that of the add-on (i.e. the MVN actor's name is "MVN System 1" and there is already an object called "MVN: MVN System 1" that exists in Blender when the stream is started). Although this would be a rather specific scenario for a user to encounter, this would still mean a loss of user data, so this should be made clear.
- d. When opening/creating a new file, the user should stop the streamer/recorder first before doing so to prevent any issues. Otherwise, Blender may require a restart for the add-on to function properly again.
- e. "Character Meta Data" and "Scaling Data" are required options in MVN's network streamer for the add-on to be able to stream properly.

Retargeting

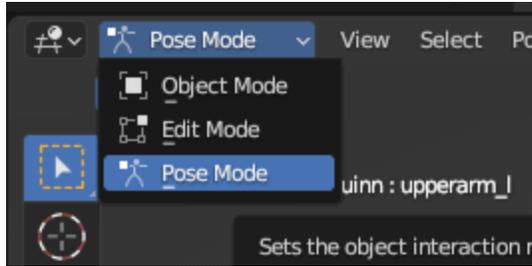
1. Clicking the text box next to “Target :” will reveal a dropdown menu of all non-MVN armatures detected in the scene. Select the desired target armature.



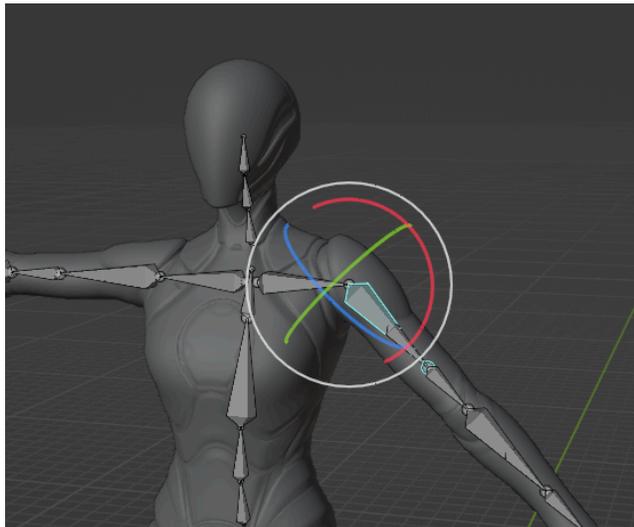
2. Clicking the “Source :” text box to choose an MVN armature to act as the source for the target.



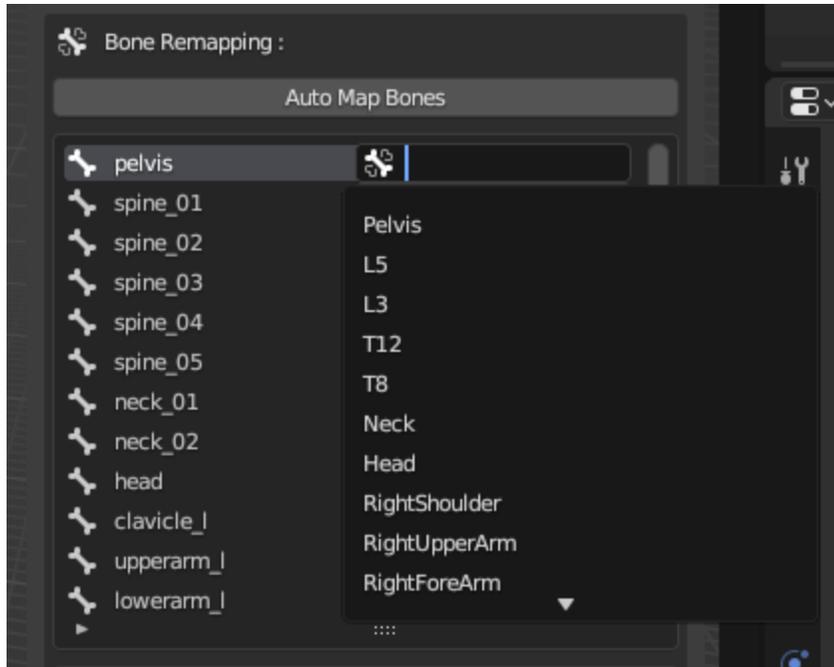
- NOTE: All armatures in the scene will appear in the source dropdown menu. This is for the purpose of offline retargeting explained later.
3. Before retargeting can occur, the target character needs to have a T-pose applied to the add-on (same T-pose used in other plugins, including for fingers). The T-pose is referenced from the target armature’s current pose in Blender’s “Pose Mode”.
 - a. If the armature is not already in a T-pose, select the target armature and go into “Pose Mode” by changing the mode in the top left corner of the viewport or by using the Blender default hotkeys “ctrl + tab”.



- b. The armature can then be rotated as needed. Rotations do not need to be applied (applying pose in Blender) or keyed. However, once the T-pose is applied in the add-on, any changes made to the target armature in "Pose Mode" will add offsets to the retargeting.



- NOTE: If there are keyframes on the bones, any rotations made by the user will be lost if they are not keyed as well or if the previous keyframes are not deleted.
4. Once the target armature is in the desired T-Pose, click "Apply T-Pose". The T-Pose can be edited and reapplied at any time.
 - a. Minor adjustments can be made to the retargeting to help with specific issues (such as the arms clipping into the body). However, if large adjustments need to be made, it is better to do these by editing the T-Pose and then reapplied.
 5. Once the T-Pose is applied, "Bone Remapping" becomes available. The left side shows all bones in the target armature and the right side contains text boxes for selecting the MVN source armature bone to map to.



- a. “Auto Map Bones” uses default naming conventions to attempt to automatically map MVN source bones based on the target bones’ names.
 - i. Naming conventions used: Xsens, Blender, HIK, Unreal, 3DS Max/Biped
 - b. Bone mappings can be saved and loaded using the buttons below the bone mapping section. Mappings are saved as .csv files.
 - i. NOTE: The default location set up by the add-on for saving and loading mappings is a folder created in the user’s AppData folder. If the add-on is removed (not just disabled), this folder and its contents will be deleted along with the add-on files.
 - c. NOTE: If the “Source” changes to an MVN armature that does not have the same bones as the previous source (e.g. switching from a source with fingers to one without), any bones mapped to those missing fingers will be cleared from the bone remapping. Users should be sure to save mappings as needed to prevent wasted work.
 - d. NOTE: Automapping or loading a bone map will overwrite any manual changes in the bone remapping.
6. Once all bones are mapped properly, the target armature should be driven by the source through FK retargeting.
 - a. The target armature can be repositioned as needed in “Object Mode”.
 - b. FK retargeting is achieved through a combination of transformation constraints and custom drivers in Python which are added to each bone in the target armature’s hierarchy depending on what was configured in the bone mapping. Each transformation constraint copies the respective bone’s rotation (with the exception of the pelvis which has an additional constraint to also track location).

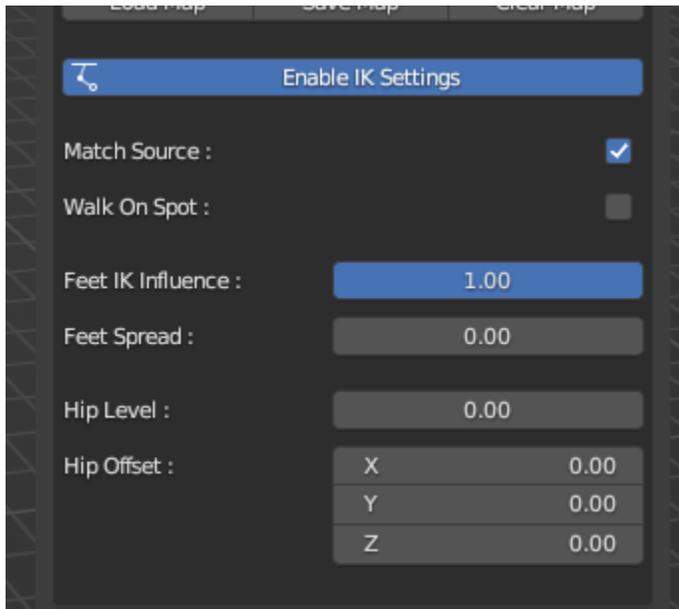


7. The FK retargeting constraints can be reset at any time by remapping bones, toggling the applied T-Pose, or resetting the retargeting source. This will only clear out constraints following the naming convention used by the add-on (“MVN_”), so user created constraints will be left alone as long as they do not follow this naming convention.
8. NOTE: For finger retargeting, if there are no metacarpal bones in the target character’s hands, the finger bones will require an offset aside from just the finger T-Pose. This behavior is also consistent from experience with other plugins, and seems to be due to the converted rotation data from MVN for the fingers being affected by the metacarpal data since the fingers are children of the metacarpals whereas that data is lost in retargeting due to not having metacarpals. One workaround for a user would be to add dummy metacarpal bones to their target hands for bone remapping. These bones would not need to be skinned/weighted to the mesh, but they would need to be parents of the respective finger bones.

IK Settings

Aside from basic FK retargeting, the add-on also has the feature of IK retargeting for the feet. For this feature to work, the target needs the following bones remapped: Pelvis, RightUpperLeg, RightLowerLeg, RightFoot, RightToe, LeftUpperLeg, LeftLowerLeg, LeftFoot, LeftToe.

1. Enable IK feet retargeting by clicking “Enable IK Settings”. This will create a new armature as well as reveal additional settings in the add-on ui.
 - a. NOTE: IK retargeting will not work for armatures without toe bones mapped. If the character does not have toe bones in its armature, simply adding dummy bones as children to the character’s feet will be sufficient for IK retargeting to function.
 - i. The IK retargeting armature does use the position of the toes as a reference for the ball of the foot in order to determine the foot’s contact point during a toe roll. If using dummy bones, it’s recommended to place these where the user would want the “ball” of the foot to be.
 - b. NOTE: The additional armature is used to determine the mechanics of the IK solving and serves as a reference for IK retargeting to the target armature. It is hidden by default in the viewport and should not be touched by the user. If the armature does get broken/deleted, the IK armature can be regenerated by simply disabling and enabling the IK settings.



- c. “Match Source” = When enabled, causes the target armature’s hips to follow the absolute translations of the source armature’s hips in the global X and Y axes. The global Z axis translation is scaled based on the difference in hip height between the target and source armatures. If “Match Source” is disabled, all three axes translations will be scaled.
 - d. “Walk On Spot” = When enabled, sets the target armature’s hips location in the X and Y axes to the origin of the target armature (similar to a No Level scenario in MVN), but retains the Z axis retargeting.

- e. "Feet IK Influence" = 0: no IK feet retargeting (instead uses the default FK retargeting); 1: full IK feet retargeting.
- f. "Feet Spread" = Adds a positional offset for the IK feet based on the orientation of the target armature's hips.
- g. "Hip Level" = Adds a global Z axis offset to the target armature's hips.
- h. "Hip Offset" = Adds a locational offset to the target armature's hips relative to the orientation of the target armature's hips.

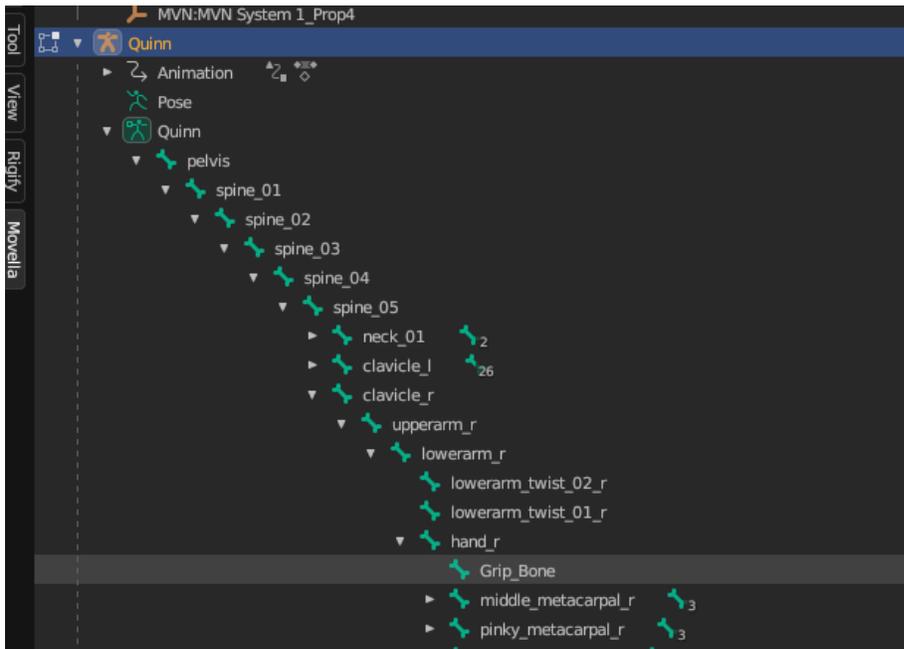
Prop/Object Retargeting

Prop and object retargeting can be achieved in a number of different ways depending on how the target object is configured.

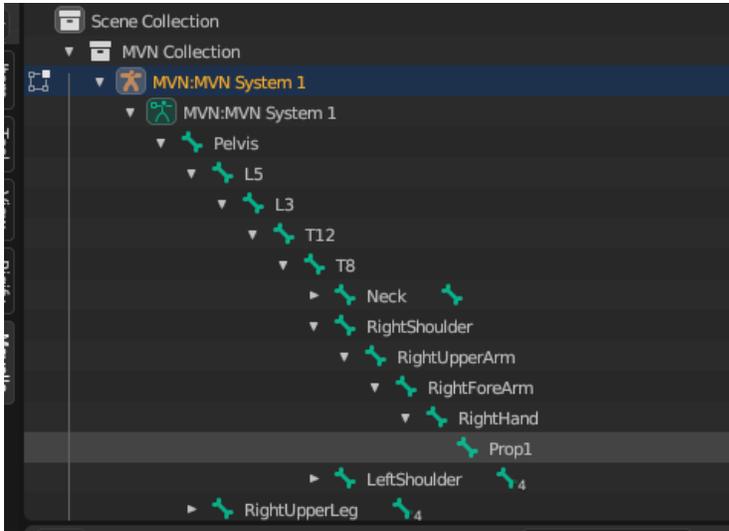
For prop retargeting, if the target object is part of the target character's armature and the desired bone to retarget to is also a part of the hierarchy in a way that matches the configuration in MVN, the add-on's built-in bone mapping and retargeting can be used. Below is an example of this scenario.

Scenario #1

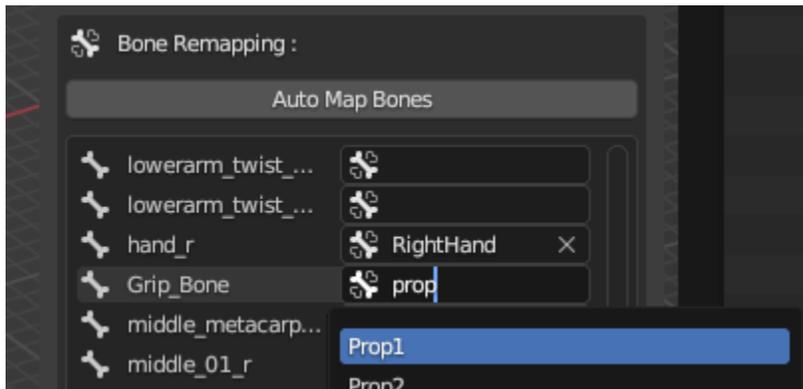
In this example, we want to retarget to a gun prop in the target armature. The bone for the gun prop (called "Grip_Bone") is part of the target armature's hierarchy and is a child of the "hand_r" bone.



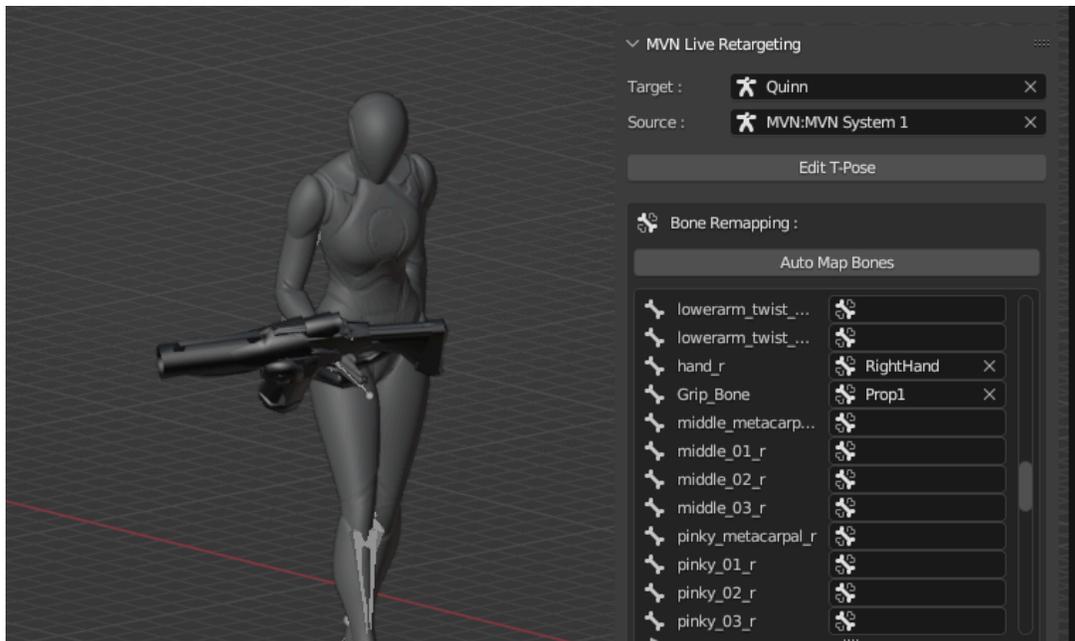
This configuration is similar to the "Prop1" bone in the MVN armature which is a child of the "RightHand" bone.



1. In the Bone Remapping, assign the correct MVN prop bone to the target prop bone.



2. The target prop bone should now be retargeted but will probably require an offset.



- This offset can be done either in the T-Pose application or after. However, since the offset is pretty significant, it's best to do this by editing and reapplying the T-Pose.



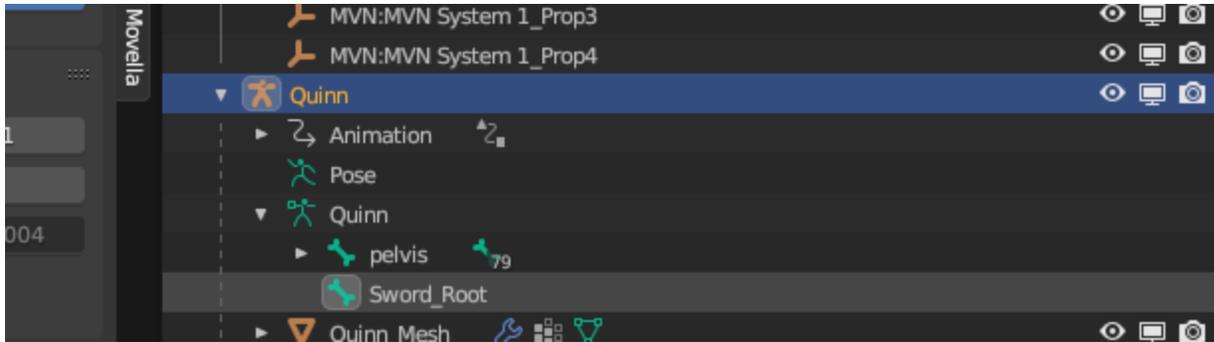
- With the offset applied, the prop should now be properly retargeted.



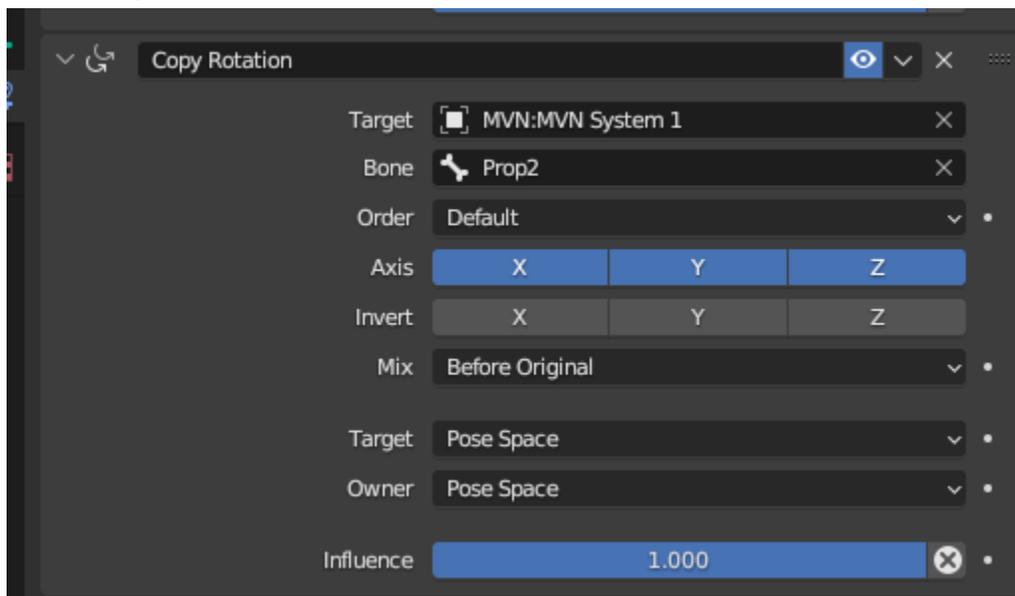
Otherwise, for both prop and object retargeting, target objects that are independent of the target character or are not included within the target character's hierarchy (i.e. part of the armature but not a child of the pelvis/hips and any bone below that) can be retargeted based on the user's own workflow. Below is one possible workflow.

Scenario #2

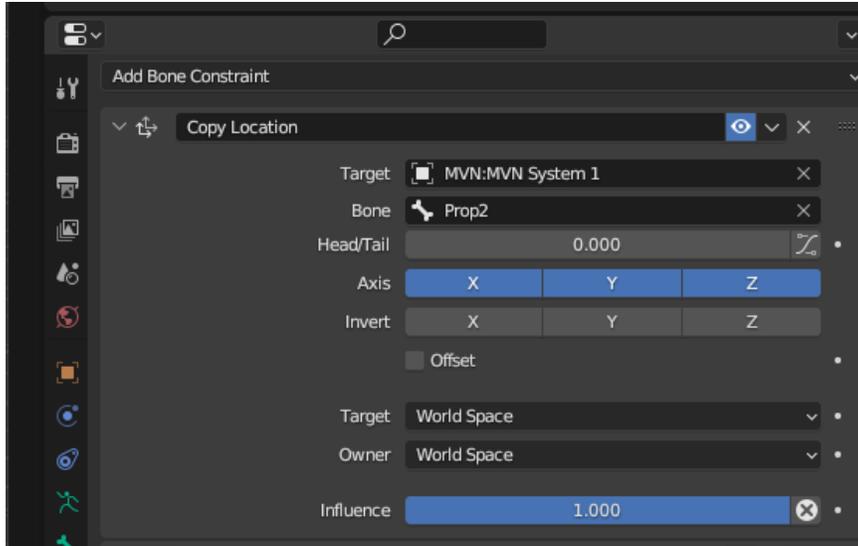
In this example, we want to retarget to a sword in the target armature. The bone for the sword prop (called "Sword_Root") is part of the target armature but it is not a child of any bone.



1. To retarget rotation data, add a “Copy Rotation” constraint to the target prop bone and have it target the desired prop bone in the MVN armature.



- Mix: “Before Original” is necessary to allow a rotational offset to be applied on top of the constraint.
 - “Pose Space” to “Pose Space” allows the user to still rotate the armature object the target prop bone is a part of without affecting the retargeted rotation.
 - NOTE: This configuration only works if going from bone to bone. In the case of object retargeting, it would always be retargeting from empty to [...] in which case it would be “World Space” to “World Space”. In this case, once the rotational offset is added to the target object, it can’t be rotated again without needing the offset to be redone.
2. To retarget location data, the simplest method is to just add a “Copy Location” constraint to the target prop bone and have it target the desired prop bone in the MVN armature.

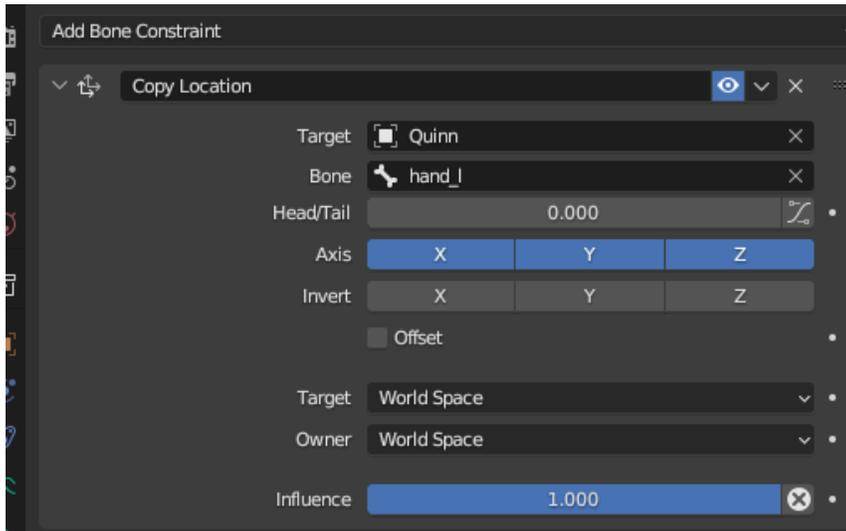


- While this method does get location data from the prop, this can cause issues when the target armature has different proportions than the MVN armature.

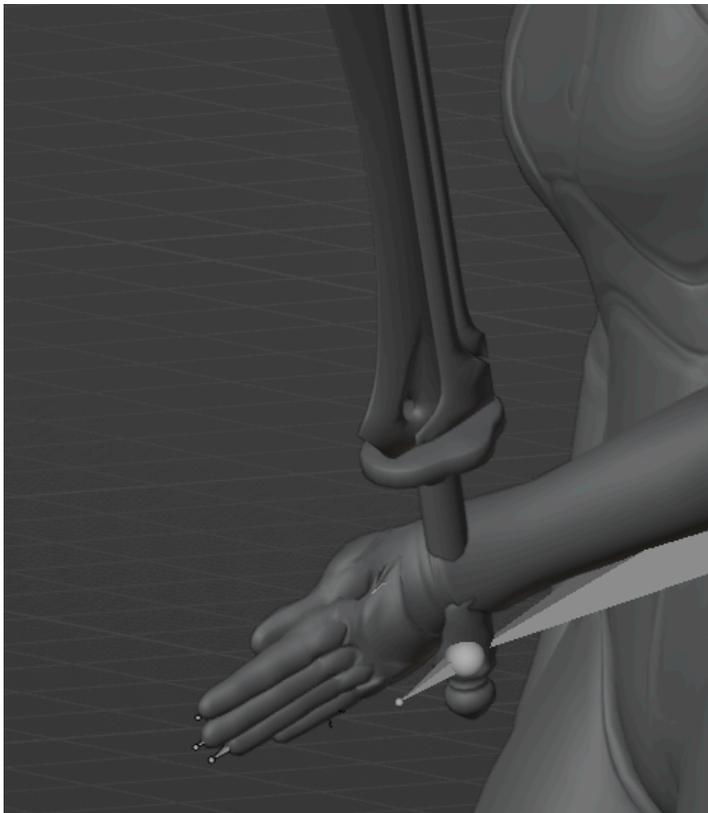


- While the sword would be lined up correctly with the MVN armature, it is misaligned with the retargeted character.
 - For object retargeting, this misalignment is inevitable since object data from MVN is absolute and not always 1-to-1 with the body data. In this case, a user would want to bake their data and manually adjust the animation.

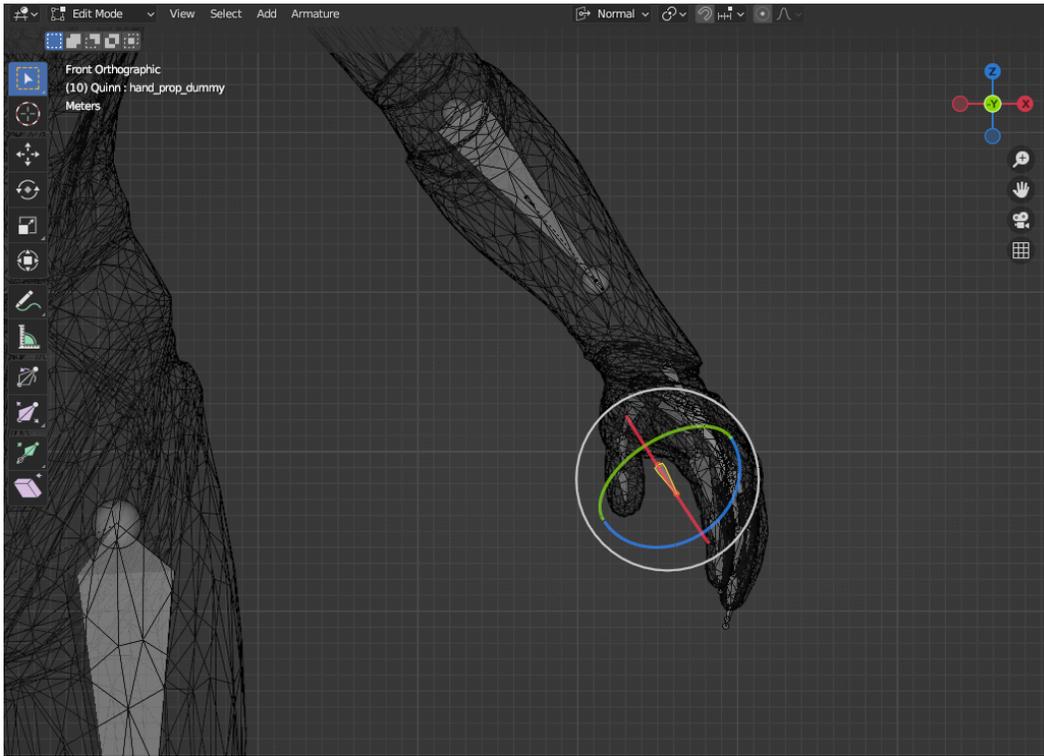
- For prop retargeting, since MVN props are parented to another segment, better retargeting results can be achieved by having the copy location constraint instead target the respective parent bone (in this example, having it target the character's left hand).



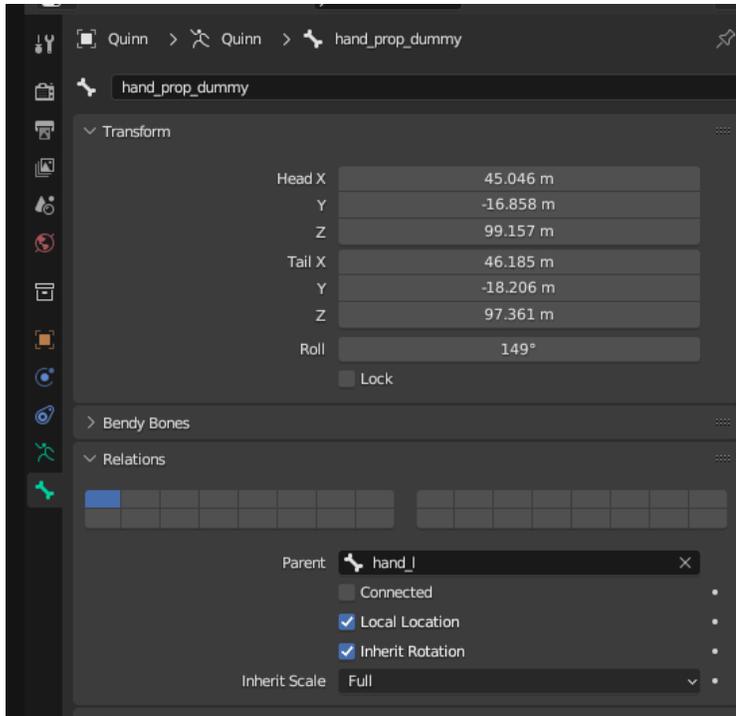
- This setup has a new limitation in that it is now copying the exact location of the target character's hand bone (specifically the head of the bone if "Head/Tail" is set to 0.000), which is usually at the wrist of the character. So the sword will maintain a consistent distance with the target character's hand but will still be misaligned.



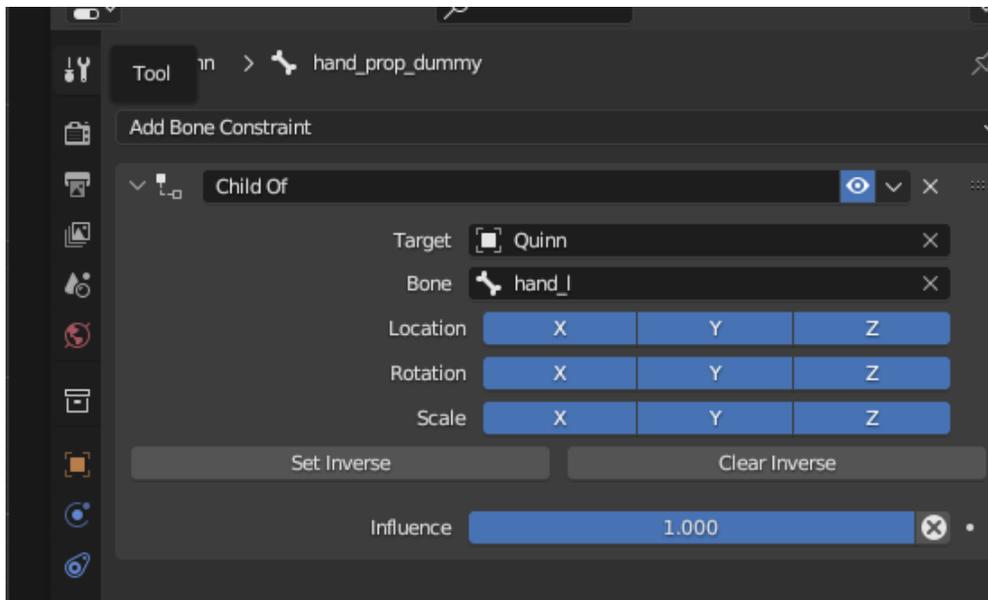
- To fix this, the simplest method would be to create a dummy bone in the target armature that can act as the location target for the sword prop. This dummy bone would then be set as a child of the left hand bone in order to maintain a proper offset with respect to the target character's hand.
- Add a new bone to the target armature in "Edit Mode". In this example, the new bone is "hand_prop_dummy".



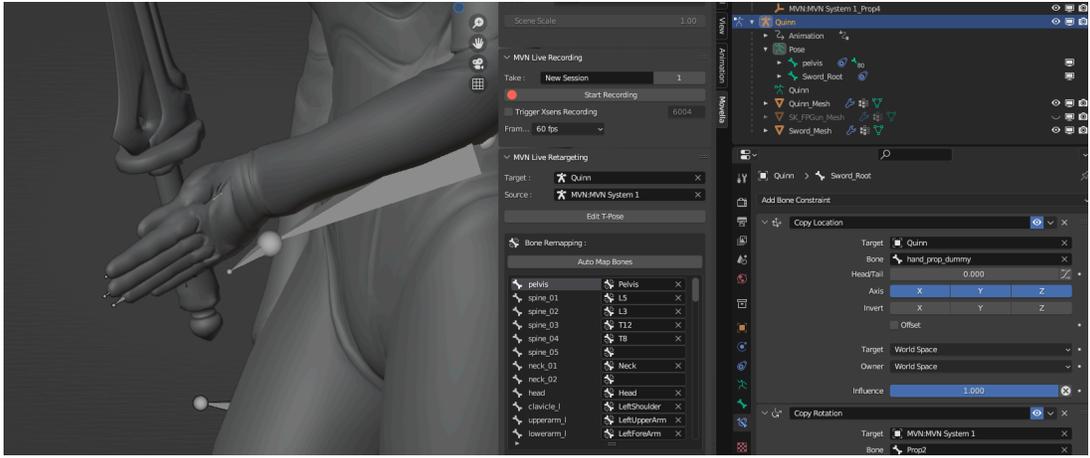
- Set the "Parent" of the bone to the desired parent bone in the target armature. In this case, "hand_l".



- Doing so adds the dummy bone within the target armature's hierarchy. If a user wanted to avoid this, they could instead add a "Child Of" constraint to the dummy bone with the target set as the desired parent bone. This is functionally the same as parenting the bone above.



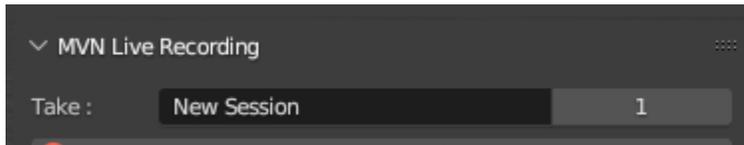
7. This dummy bone can now be repositioned as needed in "Edit Mode" for a more permanent offset or in "Pose Mode" for a temporary offset.
8. Once positioned, set the "Copy Location" constraint of the target prop bone to target this new dummy bone.



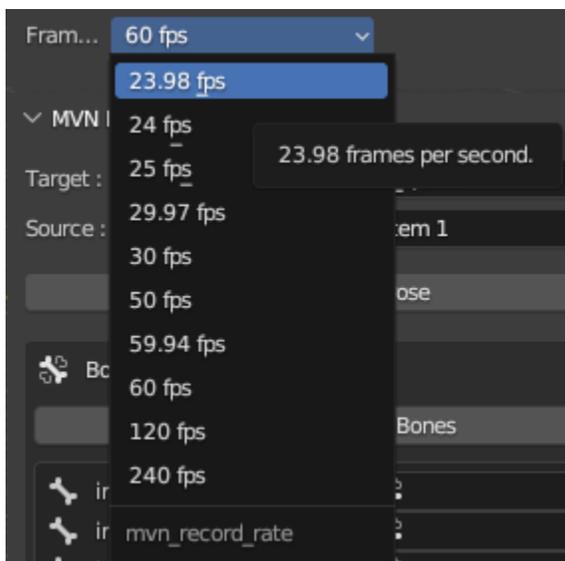
Recording

Recordings in the add-on will record any motion done by the source armatures and will save that data as actions (animation files) in Blender. These actions can then be used to drive the source armatures which can still be retargeted to a user's armature.

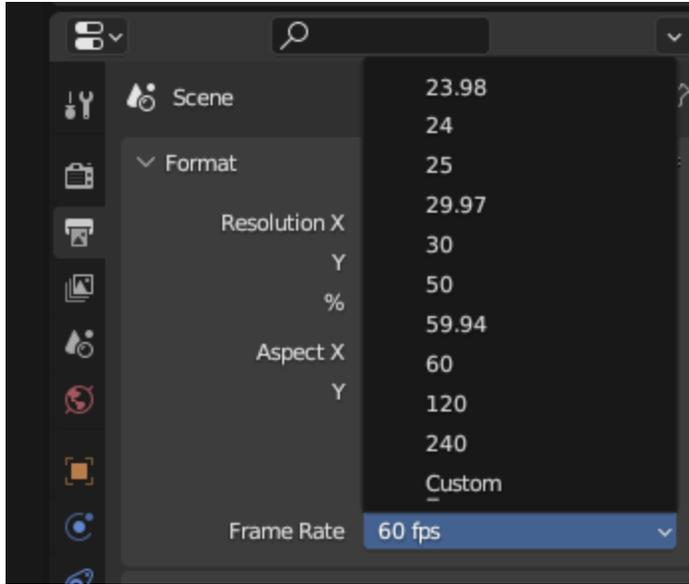
1. "Take :." will set the name of the action, and the number next to the text box will append a take number to the name.



2. The framerate for the recording can be set which determines how many keyframes will be recorded to the action.



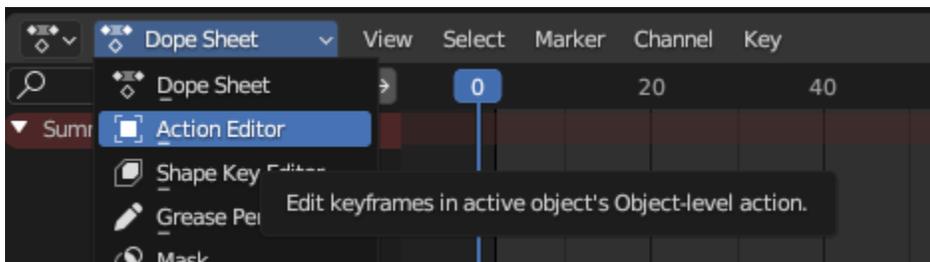
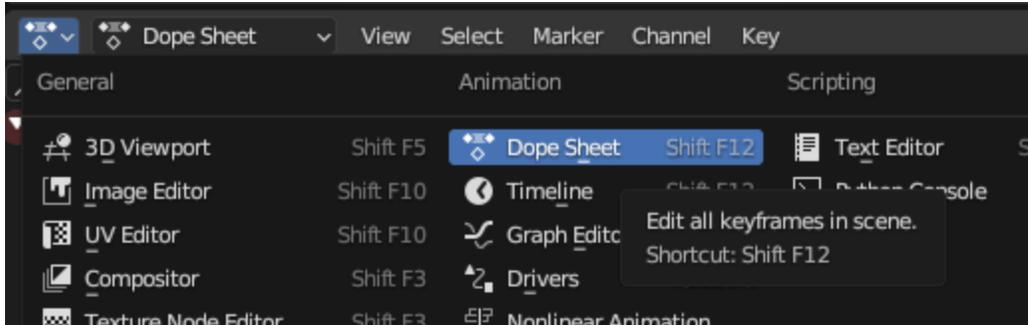
- NOTE: This setting is independent of Blender's viewport playback fps, so the recorded framerate may not appear as expected unless Blender's viewport fps also matches.



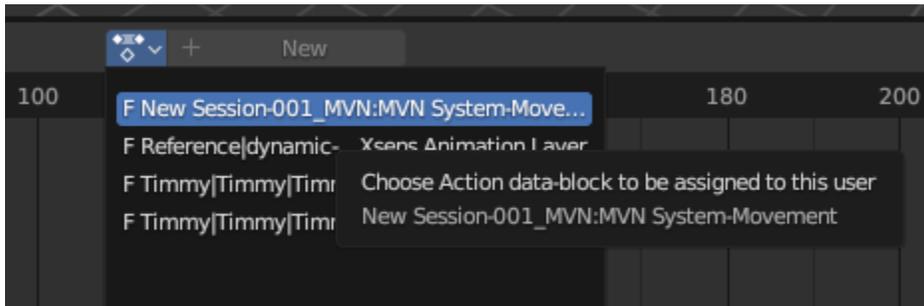
3. The add-on can trigger a recording in MVN through the UDP Remote Control function which will also control the take name and take number.

Recording Playback

Actions can be accessed by changing a window to the “Dope Sheet” and then changing the type to “Action Editor”.

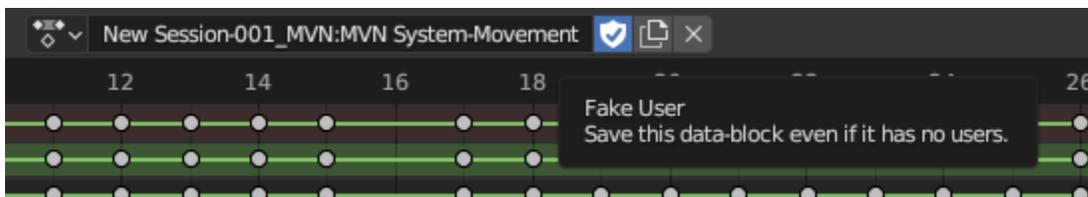


All actions in the Blender file can be accessed in the dropdown menu.



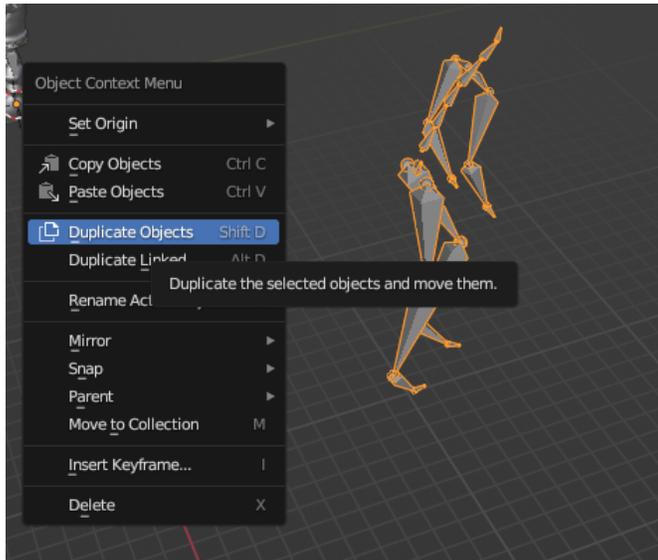
- NOTE: Actions in Blender are not restricted to any one object. Keyframes save f-curve data that references a specific object or bone name. Because of this, any action recorded through the add-on can be added to any armature created by the add-on, or even a user created armature provided the bones follow the same naming structure.

Actions will be created with “Fake User” automatically enabled to prevent them from being purged by Blender.

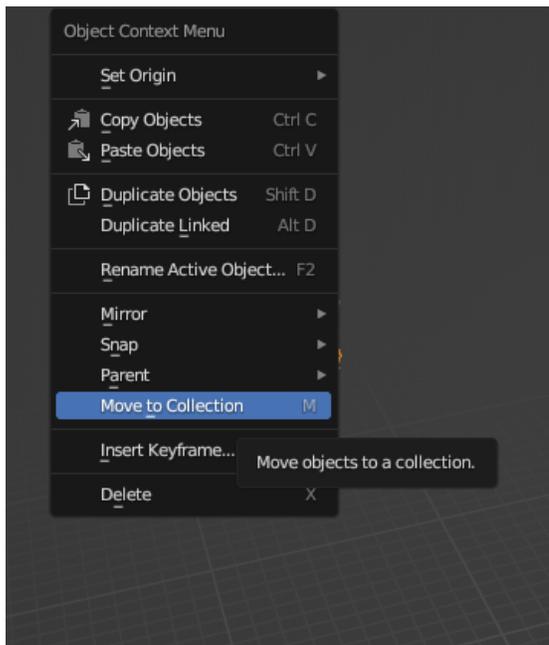


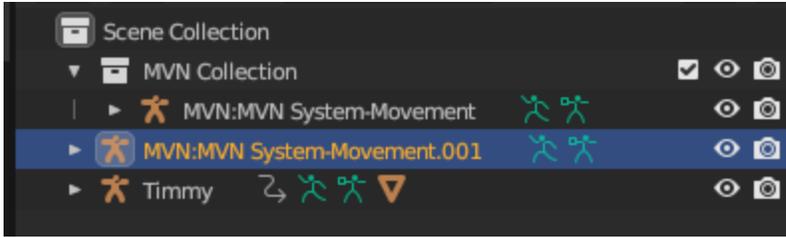
In order to playback recorded data on the correct MVN armature (so that proper scaling is maintained) while also not interrupting the live stream, the best method would be to duplicate the respective MVN armature and then assign the action to that armature.

1. In “Object Mode” in the viewport, select the desired MVN armature, right click, and choose “Duplicate Objects” or use the Blender default hotkey “shift + D”.



2. Move the duplicate armature outside of the MVN Collection to prevent it from being flushed by the add-on during normal operation. This can be done by right-clicking in the viewport and choosing “Move to Collection” (default hotkey “M”) or by clicking and dragging it to another collection in the Outliner.

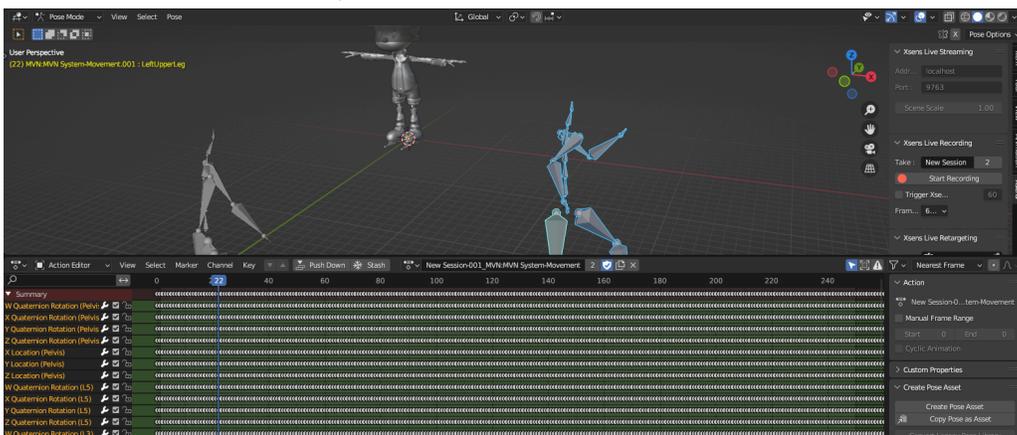




3. If desired, the armature can be renamed.
4. Navigate to the Action Editor, and with the new armature selected, choose the recorded action in the dropdown menu.



5. The action should now be driving the armature. Switching to “Pose Mode” and selecting bones should show the keyframes from the action in the “Action Editor”.

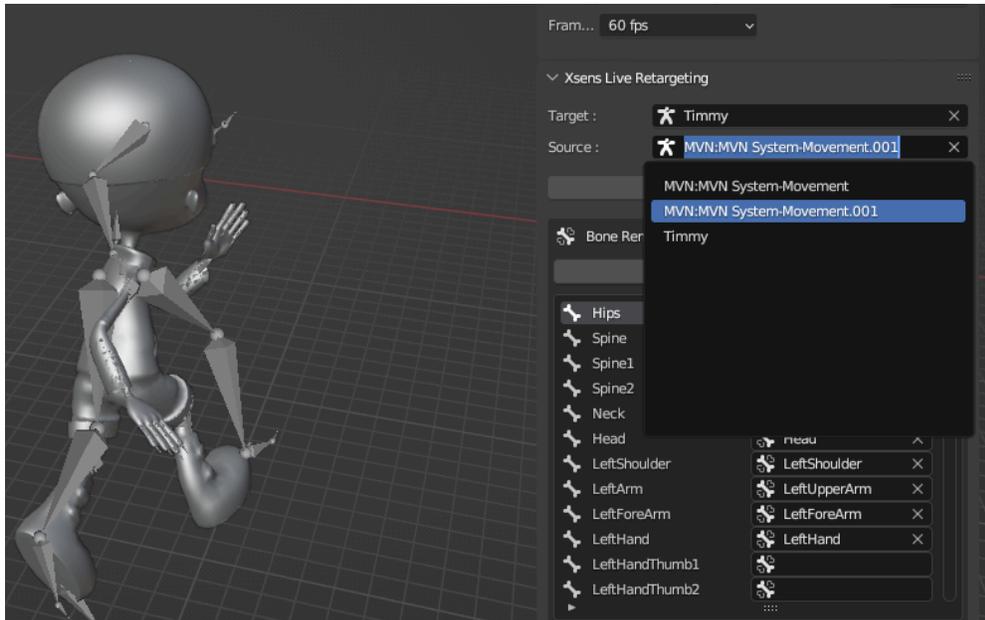


Offline Retargeting

The add-on retargeter can also be used to retarget recorded actions as well as data exported from MVN.

Scenario #1

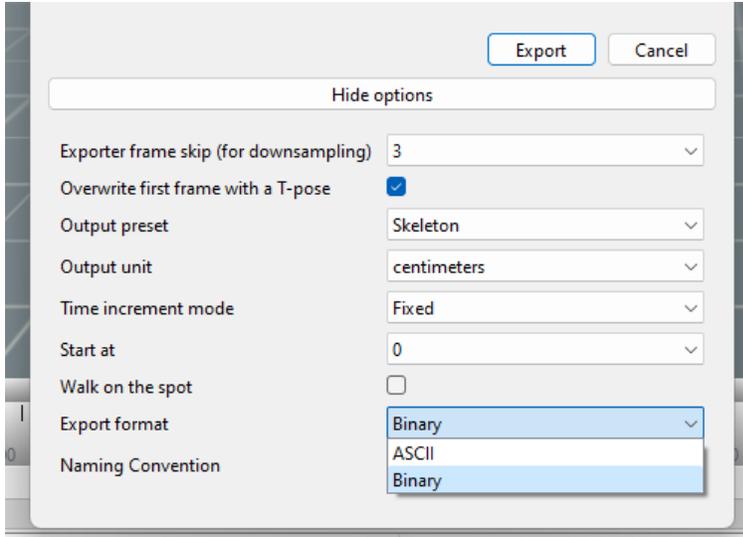
For retargeting data recorded in the add-on, the same retargeting process explained above for live streaming can be used with the “Source” armature set to the one we duplicated and assigned the action in the previous step.



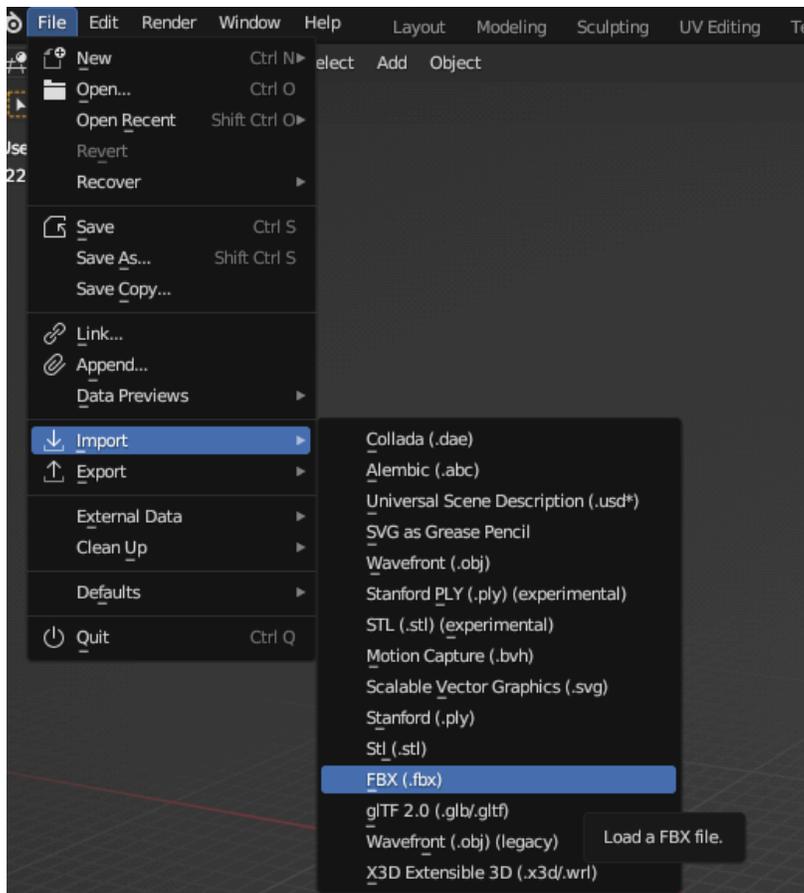
Scenario #2

For data exported from MVN, the process requires a few extra steps and comes with certain limitations.

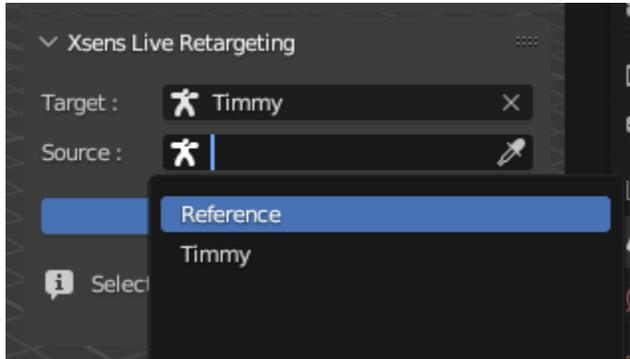
1. When exporting from MVN, be sure to set “Export format” to **Binary** as Blender cannot load ASCII files.



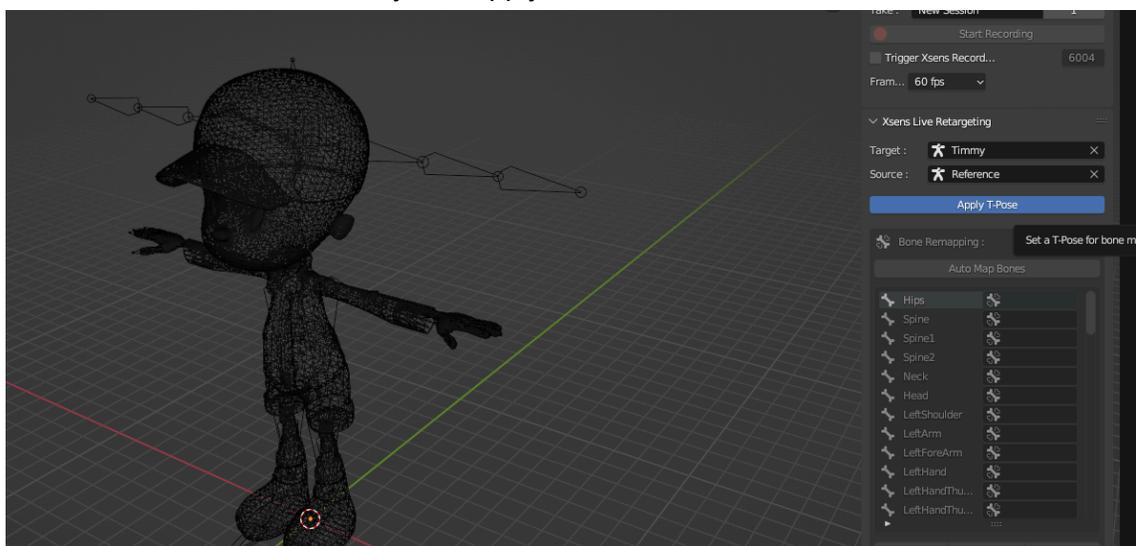
- To make retargeting easier, it's also recommended to enable “Overwrite the first frame with a T-pose”. However, this is not strictly necessary as a user could manually keyframe a T-pose in Blender later.
2. To import an FBX into Blender, go to **File > Import > FBX**.



3. In the “Armature” settings, it is recommended to enable “Force Connect Children” and “Automatic Bone Orientation”. Although it is not necessary, it makes the imported skeleton easier to work with and look much cleaner.
4. In the retargeting window, assign the “Target” to your target character and “Source” to the skeleton that was just imported. In this case, “Reference”.

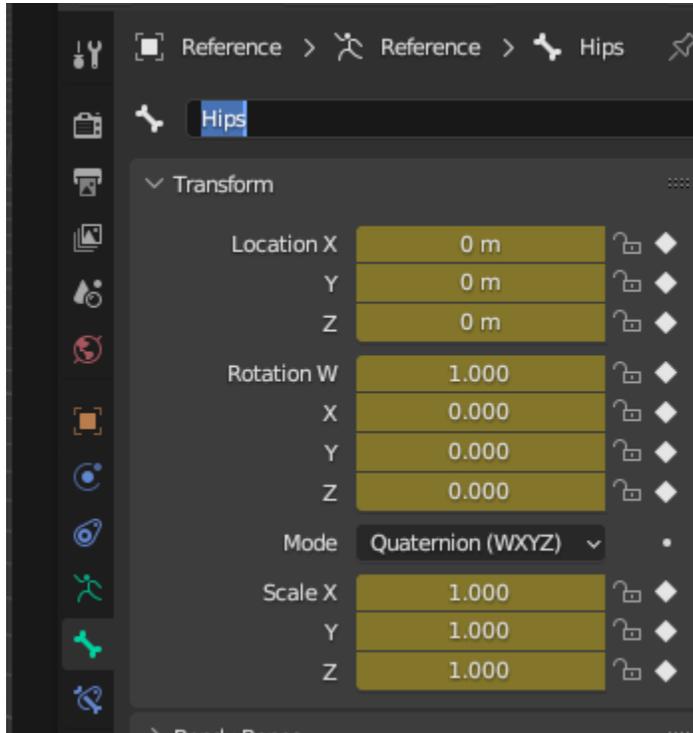


5. Both the Source and Target armatures should be posed into a T-pose if they are not already.
 - a. NOTE: Both armatures need to be keyframed in a T-pose with the timeline set to that frame when applying the T-pose in the next step. If the Source armature was exported from MVN with first frame as T-pose, this first frame should be what the timeline is currently set to. Otherwise, a user can manually set the Source armature into a T-pose but will need to be sure to keyframe that pose so that any animation data does not overwrite the pose. For the purpose of applying the T-pose, a user could also create a new action for the armature and key the T-pose there or simply unassign the action from the armature and then pose the armature.
6. Once both armatures are ready, hit “Apply T-Pose”.

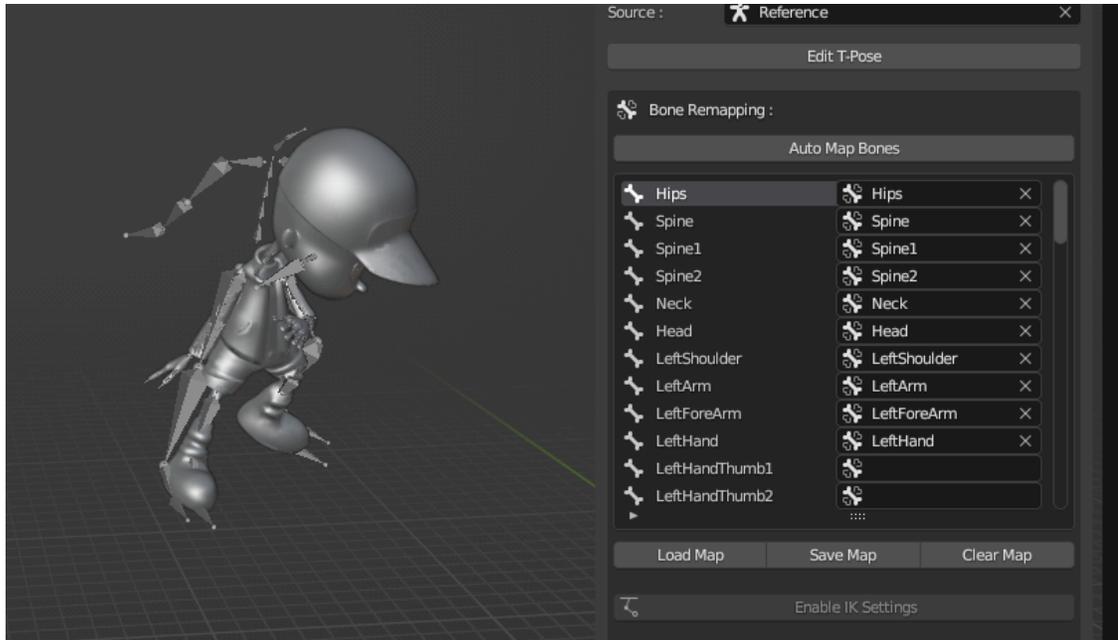


7. For bone remapping, auto-mapping will not work since the Source armature no longer uses the naming convention used by MVN armatures created by the live stream, so manual remapping will be required.

- a. NOTE: The Pelvis bone in the retargeting is unique in that it also has a constraint to copy location. Currently, the retargeter is identifying this bone by name (either “Hips” or “Pelvis”), so if the source bone that serves as the armature’s hips does not have this naming, it will need to be renamed.



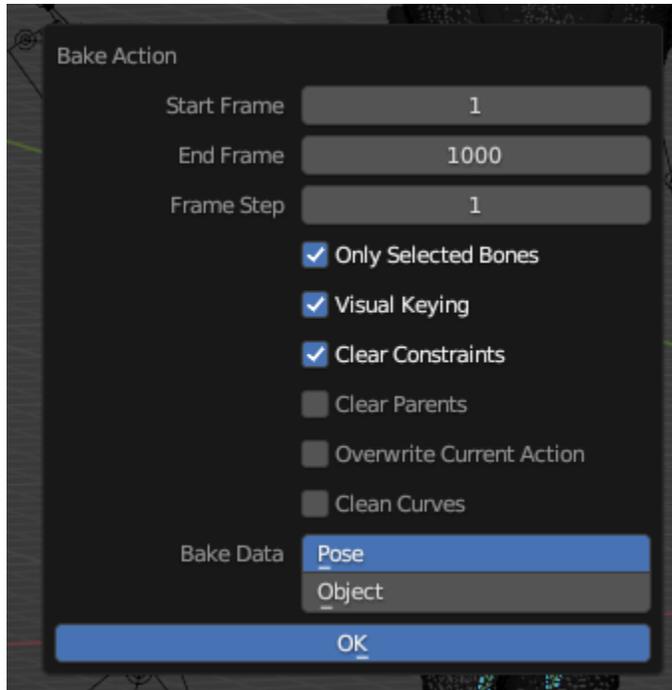
- b. When finished, the bone mapping can still be saved as a .csv as before. This is mapping bone name to bone name, so if either reference changes, the mapping file will need to be updated.
8. With the bone remapping set, the constraints should be created and the character should now be retargeted through FK.



- NOTE: Using a Source armature not created by the live streamer will cause the IK Settings to be disabled. Only FK retargeting is available for imported data.

Baking to Character

1. To bake recorded and retargeted data onto a character, select the target character's armature and change the viewport to "Pose Mode".
2. Go to **Pose > Animation > Bake Action....**



- a. "Only Selected Bones" will add keyframes to all bones selected in the target armature in Pose Mode and will exclude unselected bones. It is recommended to enable this option and select only the bones used in the retargeting, especially for armatures with a lot of extra bones.
 - b. Enable "Visual Keying" to keyframe the pose bone data as if the constraints from retargeting were applied.
 - c. "Clear Constraints" is not required to bake the pose data, but the animation will not playback correctly unless all the constraints are either deleted or disabled (the data itself will be correct, but will not look right visually since the constraints are still active).
 - i. If the constraints are cleared, retargeting will need to be set up again to restore the constraints, which can be done by toggling the T-pose in the retargeter (NOTE: Both armatures still need to be in T-pose during the apply).
 - ii. If multiple actions need to be baked onto the same character, it is best to leave this option disabled and only clear the constraints once all actions are baked to avoid having to reset the retargeting T-pose every time.
 - d. Set "Bake Data" to **Pose**.
 - e. All other settings are up to the user's preference.
3. Once finished, a new action, called "Action" by default, will be created using the target armature's bone data as reference.