Table of Contents

Global Optimization Solving for Motion Capture

The process for solving Motion Capture Data on to a skeleton involves defining a relationship between the motion capture data and the skeleton it it to drive, much in the same was as a puppeteer controls the motion of a puppet with strings. The points where the strings are attached on the puppet and the length of the strings effect the ability of the puppeteer to control the puppet and ultimately will affect the performance that the puppet is able to perform. If care and time is taken to create the correct relationships, subtle details in the performance can be recreated.

The motion capture data is usually defined as a set of global positions for the point cloud markers and the skeleton is a hierarchy of transforms (joints/segments). As the point could data moves, the skeleton's joints need to be rotated to reflect the change in the data and providing a recreation of the performance that was captured. The motion capture solver should recreate the performance as accurately as possible maintaining the fidelity of the performance.

PeelSolve uses a mathematical technique known as Global Optimization to achieve this task. Initially a relationship between the global motion capture data and the local transform of the skeleton is provided in the form of special locators that are parented to the joints to define the local positions of the motion capture markers; i.e. the position of the markers in relation to the joint/segment they are to drive. Then as the motion capture data moves, the solver rotates the joints in such a way to eliminate the gaps between the markers parented to the joints (local) and the global position of the motion capture data. The process calculates all joints and all markers at the same time using a error-reduction algorithm. This gives the potential for a very high fidelity and fluid looking result.

Solver as a Maya Plugin

Having the Global Optimization solver as a plugin for maya greatly simplifies the process by removing the need to export to another package for solving and import back in to maya again.

This allows for...

- Support for all joint attributes
- Faster iterations on improving the solve result
- Ability to see changes in the solve on more advanced rigs including Maya's constraints, plugins and skinning.
- Access to Maya's advanced interface and tools including mel/python, image planes, fcurve editor and rendering.
- Simplified process for moving joint centers to see the effect on the solve (no need to export the model back in to maya if joint centers are moved in solve package)

Installation

PeelSolve consists of a plugin for maya and a set of icons. The installation process copies the files to your computer in a location where maya can find them. The icons and mll files can be moved. No scripts are required to be sourced.

Windows

The installer will append your Maya.env file with the location of the plugin and icons. To start using the solver, load the PeelSolve[Release].mll in the Maya Plugin-Manager.

The installer adds the following entry to your Maya.env file:

```
# PeelSolve
XBMLANGPATH=C:\Program Files\PeelSolve2\icons
MAYA_PLUG_IN_PATH=C:\Program Files\PeelSolve2\plugin\2010
```

If you have other XBMLANGPATH or MAYA_PLUG_IN_PATH's defined, you may need to edit your maya.env so there is only one definition for these. The XBMLANGPATH is required for the icons to display on the shelf. The MAYA_PLUG_IN_PATH definition should add the plug-in to the Plugin Manager Window in Maya.

The uninstaller will delete all the files that were added to the system, and remove the line starting with “#PeelSolve” from Maya.env and the following two lines after. If you edit the Maya.env file you should remove the “#PeelSolve” comment.

Mac OSX

The downloaded file comes as a disk image which contains an installer.

The installer will copy the plug-in (bundle) files to the following location:

```
/Users/Shared/Autodesk/maya/[version]/plug-ins/
```

and the icon files to:

```
/Users/Shared/Autodesk/maya/[version]/icons/
```

Maya will should to find these directories by default when started.

Linux

Uncompress the archive and follow the instructions in the readme file. The icons and mll files will need to be copied to a location where maya can find them. An install script is included to help.

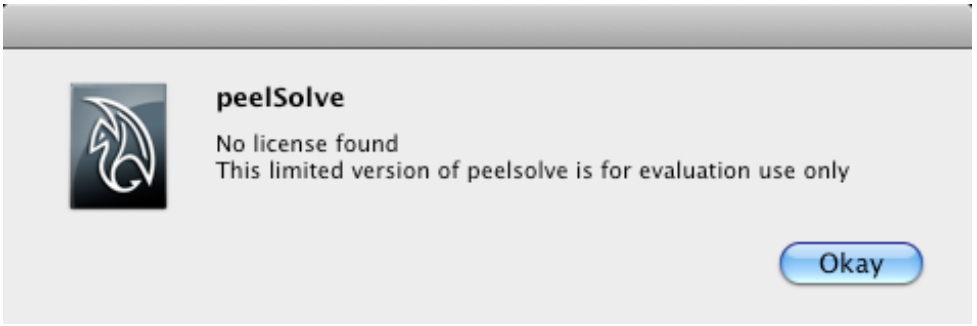
License

For the solver to operate a valid license file must be found, or the software must be run during the Beta period it has been intended.

To get a list of valid license file locations, load the plugin and run the mel command “peelSolve -lic”.

No license found

If a license file cannot be found this message will appear when first loading the plugin:



This will result in not all of the functionality of the solver being available. Some joint attributes will be missing (such as stiffness) and only position constraints are available. This allows the software to be used for evaluation and testing purposes.

Evaluation License

If you have an evaluation license the following dialog will appear when loading the plugin:

The software will be fully functional, but may only be used for evaluation or testing purposes.

Solver Operation

To confirm the solver is operating, you should have a PeelSolve2 drop down menu. The “About” option on this menu will display the version number and license status, as well as the mel command “peelSolve -v”

Setting up a solve (Quickstart)

To run a solve, at a minimum the solver needs to know of the relationships between markers and the skeleton and the root node of the skeleton which will receive the root translation and rotation needs to be identified.

Here are the steps to set up a simple solve:

1. Pose the skeleton inside the motion capture data.
2. Select a marker, or group of markers, and lastly select the bone they are to drive (the selection process is similar to creating constraints in maya).
3. Click the 

shelf icon, to indicate the position of the markers are to be used to drive the bone.
4. Select the root node of the skeleton. This should be located at the hips and should be the parent of all other joints.
5. Click the 

shelf icon, then click on “Add” on the window that comes up to tell the solver that this is the root node of your system.

At this stage, all of the markers that were created and parented in to the skeleton (spheres with lines drawn to the parent) should be in exactly the same place as the motion capture data they are being driven by. Now if you make a small change to the motion capture data, like to move a few frames forward, or select a motion capture marker and move it, we can get the solver to accommodate for that change.

Running the Solver

To run the solver for the current frame, click the 

icon. The solver will search for the pose that brings together the global markers (mocap data) and local markers (parented to the joints). The joints will be posed only, no keys will be saved or modified.

To run the solver for a range of frames, click the 

icon. The solver will solve each frame and set a key for the pose.

This is the basic steps for setting up a solve. From here the process can be refined

refined to provide a higher fidelity or better looking result by altering the relationship between the skeleton and the motion capture data, and modifying the attributes that affect the way the solve is calculated.

Tuning the Solver

The relationship between the markers and skeleton is vital to achieving the best looking and highest fidelity result. While solver attributes can be adjusted, the most effective way to fix problems and improve results is to change the way the markers relate to the skeleton. If the system is being forced in to a difficult position, the solver may have a hard time finding a solution and artifacts such as popping or unnatural orientations and movements can occur.

In particular, the position of the joint centers in the system can make the difference to the result. Unfortunately is is not always possible to change the joint centers on a production model, so the artifacts must be reduced. Even with perfect joint centers, the solver can have trouble accommodating for markers that slide around on the suit and for flexible actors in a capture subject versus the rigid joints (no translation) in a computer model. Changing the relationship between the markers and the skeleton can help reduce these artifacts.

The whole system is solved at once, so every bone is dependent on every marker, if only by a very small amount. This means that it is possible for the hand markers to pull the arm in a particular direction which pulls the clavicle, spine and root. So while problems may appear in the root or spine, the cause of the problem may be further down the chain.

Closing Gaps

The first place to look for where problems are is where there are spaces between the active markers (the spheres parented to the skeleton) and the motion capture data that they are trying to reach to. The bigger the distance there is, the more error will be in the system and the greater the chance of artifacts occurring.

On hands and feet, the markers may fall short of their target, which means the joint is not long enough at that pose, which may be because the joint center is not in the correct place. If the markers are beyond the target, then the joint may be to be too long.

Removing Offsets

Sometimes the problem may be a simple offset. If the posing of the skeleton in relation to the motion capture data needs to be adjusted, the easiest way to do this is to lock the marker's position then rotate the bones, then unlock the markers again. It is important to remember to unlock the markers, as if the bones are moved by animation or solving, this can permanently apply offsets to the rig (Undo will generally not return it to the original state).

To lock the markers, select the marker you want to lock then click on this icon: 

To unlock all markers that have been locked, click on this icon: 

Tip: You can select all the markers from the peelSolve 2 Menu → Select → Active

Transform Types

In addition to “Position” constraints, there are a number of other types of constraints, that provide alternative ways for the markers to be driven by their target.

They are all assigned the same way; select the motion capture markers then the bone it is to drive and click “pos”, “ori”, “both”, “slide” or “aim”. Markers will be parented in to the skeleton and connected to the source object. The source does not have to be motion capture markers, any object that has a world position in maya can be used.

The name of the marker does not need to be the same, nor does the source marker need to be in a particular hierarchy. The relationship is defined by creating a connection of the “worldMatrix” attribute.

If there is only one selection, it will set the selection to being an active marker of that type.

	Match on global position (world translation)
	Match on global orientation (world rotation)
	Match on global position and orientation (world translation and rotation)
	Match on position, but allow marker to slide along axis of the child transform.
	Make marker point towards target, but don't pull.

When attaching motion capture data to a skeleton, the “pos” constraint is generally the most applicable. Slide may also be useful for markers located around the elbows and knees.

If the transform that is driving has rotation information, for instance a rigidbody made up of three markers, the “ori” or “both” can be used. For instance if three or four markers for the head are turned in to one marker that has rotation, then that marker can be used to drive the head joint with a “both” or “ori” constraint.

The ability to solve based on both translation and rotation allows for solving for a wide range of setups rather than just point cloud data on to a skeleton hierarchy. It is possible to solve from skeleton to skeleton, or create alternative character rigs that go beyond the traditional motion capture setups.

Tip: When using “Both” or “Ori” markers, the orientation target is solved at the position of the marker. In many cases you want the orientation to be applied at the joint. For this reason it's generally best to remove any translation on the target marker (active marker). Also, its best to avoid using “Both” markers; in many cases using both a “pos” and an “ori” marker is more effective, as the translation on the ori marker can be removed so the rotation is solved without any offset.

Transform Attributes

There are a number of attributes that can be added to a joint that is being driven by the solver. Default values are used if the attributes are not available.

To add the attributes to a joint, click this icon: 

These attributes can help to tweak the solve and reduce artifacts. With Global Optimization solving, it is best to fix the cause of the problem at the source, which is generally is the motion capture data.

Stiffness

If a joints has too few or no markers associated with them, then the joint will be very free to move around and may create a lot of wobble or random looking results. Stiffness factors in the default pose as a preferred solution, effectively making the bone “stiffer”.

To set the default pose for a selected joint, press this icon: 

To rotate a selected joint in to its default pose, press this icon: 

Stiffness can be applied to a joints rotation, to the length degree of freedom, the joints translation. The preferred position is stored on the attributes “preferredTrans” and “preferredAngle”.

Rotation Stability

“Rot Stab” will factor in a certain amount of the previous frame in to the result, to help provide a more stable solution.

Length Degree of Freedom

“Lendof” allows a bone to translate along a particular axis, as defined by Length Axis X, Y and Z. This is particularly useful for spine bones, allowing them to grow and shrink as the torso folds.

Calibration Degree of Freedom

Calib Dof is a experimental feature which is designed to guess the position of a joint center. Please see the section on Calibration solving below.

Rotation Degree of Freedom

By locking a joints rotation channel, or turning off of of the degrees of freedom in the joint's attribute editor, the solver will lock that axis. It is important to remove any keyframes before locking.

Translation Degree of Freedom

By enabling “dofX”, “dofY” or “dofZ”, the joint will be allowed to translate along that axis, and will receive translation keys. It is important that the this is enabled for the root, and generally should not be enabled elsewhere unless it is specifically needed. If enabled, it may make the joint too flexible and yield unnatural looking results.

Rotation Sharing

Rotation Sharing is an experimental feature to allow even distribution of the rotation over a chain of bones. If a series of bones in a chain have rotation share turned on, the first should have a value of 0 and the last of 1 and the bones in the middle of the chain set between 0-1 representing the position in the interpolation. For example if you were to have four bones A > B > C > D in a chain and enable rotation sharing for them all and weighted them as 0 > 0.25 > 0.5 > 0.75 > 1; if A and an orientation of 0 and E had 90, the distribution of the rotation would target for 0 → 22.5 > 45 > 67.5 > 90.

Errors

Solver errors can be saved on to the bones, to provide feedback and for troubleshooting. The errors are provided separately for translation, rotation, stiffness and stability. To enable the errors, turn on “statistics” in the solver options.

Solving

There are a number of ways to invoke the solver, and a number of parameters that will affect how the solving is done.

	Solve current frame. This will read current values and set translations and rotations without saving any keys.
	Solves a frame range with the following solver options overriden: Method: Matrix/Quick, Iterations: 50. This will solve quickly but may not give an optimal result.
	Run a full solve for the frame range using the current solve parameters.
	Solves a frame range using the “Descent/Slow” solver and “Refine” turned on. All other solve options are retained. This should be used to re-solve a completed solution and will try to improve the result. This may be slow, but should improve the result.
	Turn on a script job. This will run the “solve current frame” any time a transform or marker weighting is changed.
	Turn off the script job.

The “Test” solve can be used prior to solving a full frame range. Scrub to a some key poses in the performance and ht the “test” button to see how well the skeleton fits in to that pose.

It is a good idea to a “Test” solve before solving a frame range, as this will ensure that the solve starts off with a good initial pose to work from. If the skeleton is far away from the marker data, the solver may have trouble finding the initial pose, particularly if the number if iterations is set to a low value.

Only turn on the script job when tuning or testing a solve on a given frame. It is mostly provided to avoid continual clicking on the “Test” button when tweaking marker positions or weighting. Always turn it off when finished and avoid scrubbing the time when it is on.

Calibration Solve

A calibration solve is an experimental feature that can be used to help determine joint centers based on the arc of movement around a joint. Only a small number of frames are usually needed, around 20 or 30 should be sufficient. The motion should move the joint that is being calibrated though a wide arc. The calibration dof should be turned on for the joint that is to be calibrated only.

A calibration solve will disable length and translation dofs for all joints and enable any calibration dof that have been set. The other solve types will ignore the calibration dof if it is set.

When the solve runs, the joint should translate towards the center of the arc it is being rotated around.

To run a calibration solve, select “Calibrate Solve frames” from the drop down menu. As this is an experimental feature, there is not a shelf icon for this type of solve.

Solving Parameters

There are a number of global parameters that affect how the solve is run. These parameters are stored in a node that is saved with the scene.

To modify the parameters, click this icon: 

This will bring up the attribute editor with the solver options node selected.

The following solver parameters can be set:

Iterations	The maximum number of steps the solver should take when working on a solve. By using a smaller value for this will make the process faster and a cost in the quality of result. (see note below)
Time Mode	Specify a time range to solve or use the time slider when solving a range
Reverse	Solve backwards, useful when the data at the start of a take is messy, for instance if the capture subject enters the volume at the start of the move
Delete Keys	Remove keys prior to solving; none, all keys or the keys in the solve frame range
Method	Choose a the mathematical method of finding the result; Matrix(Quick) – fastest method but least stable, Secant (Faster) – A good general purpose method, Descent (Slow/Accurate) – The slowest but most stable result.
Gradient Samples	The number of samples to use for gradient estimation. Increasing this value will increase the accuracy of the search method, but will slow down the amount of time for each iteration.
Debug	Output debug data to console.
Statistics	Show solver statistics after every frame and create animation curves with statistical data.

Read Direct	Read fcurve data directly rather than evaluating the scenegraph. This will speed up the solver, but will not work if you are using constraints or other connections.
Calibration Type	Changes the method of calibrating a joint center while doing a calibration solve. This parameter will not affect regular solves.
Scale	Scale the translations during the solve
Bothways	Solve Forwards, then backwards
Refine	Once a solve has completed and there are keys on the target skeleton, use “Refine” to attempt to resolve and improve the solution. The starting point for each frame is taken from the skeleton, rather than the previous result.
Rootfirst	This feature is currently disabled
Root Nodes	Specifies the root nodes to solve
Quaternions	This attribute is listed in the options under “Extra Attributes”. It will cause the solver to use quaternions for joints that have three degrees of freedom for rotations.

The solver will stop solving when it thinks it has found a solution (either by minimum variation in the error or by reaching the maximum number of iterations). By turning down the iterations it will set a limit for the number of calculations. To get an idea of how many iterations are used for a solve, turn “statistics” on and solve a frame. The “number of steps” reported in the statistics is the number of iterations used to obtain the result.

Mel Command

The core solver is implemented as a mel command “peelSolve”. This command can be written in to your own scripts.

To get the latest mel options run “peelSolve -h”.

To get information about the status of of the license, and the locations of where the license file is looked for, run “peelSolve -lic”

Here are the arguments:

-s|skel *string*	Specify a root node (may be used more than once)
-d|debug	Output debugging data
-ns|nosolve	Do not solve, just parse the scene
-lc|listChannels	List channels that the solve writes to
-lt|listTransforms	List transforms the solver affects
-la|listActive	List active transform nodes
-lp|listPassive	List passive transform nodes
-i|iter *int*	Set number of iterations
-v|version	Show version then exit
-st|start *float*	Set start frame
-en|end *float*	Set end frame
-in|inc *float*	Set increment amount
-r|reverse	Solve backwards

-stat|statistics	Show solve statistics
-m|method *int*	0: Matrix/Fast, 1: Scecant/Regular, 2: Descent/Slow
-ct|calibType *int*	1: Constant, 2: Angle, 3: Iteration, 4: Angle/Iteration
-cal|calibrate	Do a calibration solve (enable calib dof and disable len/trans dofs)
-c|cache	Cache results
-scale|solveScale *float*	Scale translations by a value
-f|file *string*	Write solve data to file for offline solving. Must be one -f for every -s flag if used
-rd|readDirect	Read fcurve data directly, rather than evaluating the scenegraph
-ref|refine	Read keys and resolve from existing keyframe for improvement
-gs|gradientSamples *int*	Number of samples to take during gradient estimation; 1, 2, 4.

Mel Examples

```
// Solve a single frame and don't write keys

peelSolve -s joint1


// return list of active joints – always use eval, not `...`

string $jointList = eval("peelSolve -s joint1 -nosolve -la");


// Solve a frame range

peelSolve -s joint1 -st 1 -en 100 -in 1


// Solve to a file for offline solving

peelSolve -s joint1 -st 1 -en 100 -inc 1 -f "c:\\joint1_solve.xml"


// select all passive joints using the roots specified by the

// options node

string $roots[] = getAttr("peelSolveOptions.rootNodes");

if(size($roots)==0) error("No skeleton top node defined");

string $ret = "";

for($i=0 ; $i< size($roots); $i++) $ret += "-s " + $roots[$i];

string $jointList[] = eval("peelSolve -lp " + $ret);

select $jointList;
```