Project 6: Twister

Due: Apr 23, 2018, 11:55PM

Have you played Twister? Your body is basically solving an inverse kinematics problem when you are playing Twister. In this project, you will build a virtual Twister game by developing an interactive IK solver. You will be given with a 3D human model with some handles defined on the model. The user can grab any handle and drag it around in 3D space via a mouse interface. As the handle moves around, your IK solver will solve for a new poses that satisfy the new locations of the handle, using optimization method. Since this is meant to be an interactive application, you need to make sure that the IK computation is very efficient. Again, you will start with the skeleton code written in DART framework. Unlike previous projects, this time you will actually use the functionality of DART, such as the data structures of articulated systems, computation of transformation chains, and the derivatives of transformations.

The requirements of this project includes:

1. Rewrite the skeleton code such that it is general to arbitrary skeleton.
2. Your IK solver must run at interactive rate (roughly 10 solves per second).
3. The IK solver should be able to solve multiple constraints simultaneously. Once a constraint is added, it will remain active until the user remove it. You might want to change the structures of mC, mJ, mConstrainedMarker and mTarget so they can hold information for multiple constraints.

Skeleton code

You will use the skeleton code provided in DART to complete this project.

Unzip CS7496P4.zip and put the twister folder in <dart_install_dir>/apps/
Put human.skel in <dart_install_dir>/data/skel/
Put Geometry.h & Geometry.cpp in <dart_install_dir>/dart/math/ to override the old copies
Put remaining *.h & *.cpp files in <dart_install_dir>/dart/dynamics/ to override the old copies
(Mac/Linux)
Remove <dart_install_dir>/build directory and create a new one.
In the new build directory, cmake.. and make
(Windows)
Re-run CMake, reconstructing the <dart_install_dir>/build directory.

UI control provided by the skeleton code:
Alt + Left-Click on a marker: create a constraint and drag to modify the target location.
Alt + Right-Click on a marker: release the constraint.
Left click: rotate camera.
Right click: pan camera.
Shift + Left click: zoom camera.
The skeleton code provides a partial IK solver which only solves for the pose of the right leg. If you try to grab the handle on the right foot by Alt + Left Click, you can see that the right ankle, right knee, and ight hip are trying to satisfy the desired location of the handle, but the rest of the body is not moving. Grabbing other handles has no effect currently in the skeleton code. If you look into MyWorld.cpp, you will see how the right leg is solved. However, the current updateGradients function is hard-coded only for the right-leg-only case. To create a general IK solver, you need to rewrite multi-layered loops such that it handles arbitrary skeleton with arbitrary number of body nodes, joints, and degrees of freedom.

As MyWorld is instantiated, it loads in a human figure as dynamics::SkeletonPre mSkel. Going through MyWorld::solve() and MyWorld::updateGradients() helps you understand how to use DART to build an IK solver. This list of How-to's can also help you understand the part of DART essential to this project.

FAQ

How to get the local coordinates of marker i?
```
Marker* mark = getMarker(i);
Vector3d pos = mark->getLocalCoords();
```

How to get the current world coordinates of marker i?
```
Marker* mark = mSkel->getMarker(i);
Vector3d pos = mark->getWorldCoords();
```

How to access the bodynode where marker i resides?
```
Marker* mark = mSkel->getMarker(i);
BodyNode* node = mark->getBodyNode();
```

How to get the the joint between bodynode, bn, and its parent?
```
Joint* jt = bn->getParentJoint();
```

How to get the the parent bodynode of bodynode, bn?
```
BodyNode* node = bn->getParentBodyNode();
```

How to get the number of dofs in a joint, jt?
```
int nDofs = jt->getNumDofs;
```

How to get the transformation chain from the root to bodynode bn?
```
Matrix4d mat = bn->getTransform().matrix();
```

How to get the transformation chain from the parent of bodynode bn to itself?
```
Matrix4d mat = bn->getTransform(bn->getParentBodyNode()).matrix();
```

How to get the transformation for j-th dof in a joint jt?
```
Matrix4d mat = jt->getTransform(j).matrix();
```

How to get the derivative matrix wrt j-th dof in a joint jt?
```
Matrix4d mat = jt->getTransformDerivative(j);
```

How to get the global index of the j-th dof in a joint jt?
```
jt->getSkeletonIndex(j);
```

How to get the transformation from the parent bodynode of a joint jt to itself?
```
Matrix4d parentToJoint = jt->getTransformFromParentBodyNode().matrix();
```

How to get the transformation from the joint jt to its child bodynode?
```
Matrix4d jointToChild = jt->getTransformFromChildBodyNode().inverse().matrix();
```

How to get the root bodynode of a skeleton, skel?
```
BodyNodePtr root = skel->getRootBodyNode();
```

Extra points

1. Add an objective function G(q) that tries to match the initial pose (when the model was loaded in) (2 points).
2. Implement adaptive step size α (1 point)
3. Make sure that the knees never bend backward (1 point)
4. Create a constraint so the left hand always holds the right hand (1 point)