Layered Depth Image Pipeline Documentation

François Faure, Sébastien Barbier, Jérémie Allard, Florent Falipou

April 6, 2011

Abstract

This is a brief documentation on the Imaged-based Detection and Response between Arbitrary Volume Objects[**?**] implementation done in Sofa. Please report to the paper to get all the technical details of the method itself. This documentation is about how this Collision detection and response has been created using Sofa, and how to use it.

1 Documentation

1.1 Overview

The method is designed to work with manifold triangular meshes. Please refer to the paper[**?**] for detail on how it works. In some cases, non-manifold meshes can behave well: for example, a flat floor described by two triangles can be used, if the normals of these triangles define an infinite half-space.

1.1.1 Sofa Components

In term of Sofa Components, this package includes:

- **LayeredDepthImagesPipeline** or **LDIPipeline**: Replaces the default collision pipeline used in Sofa. This is the only specific component you you need to define. Everything else in your scene remains the same. It will modify the ContactManager to set the "default" response to "LDI", and trigger the common methods used to detect and do the collision response. These methods are implemented in **LDIDetection** (see the Collision Pipeline section). The parameters available to configure this component are:
  - **selfcollision**: Activates the self collisions. Warning, you have to activate them in the TriangleModel too to be taken into account.
  - **pressure**: The uniform pressure force applied at each contact found between a pair of TriangleModels.
  - **pressureself**: The uniform pressure force applied in response of a self contact. Generally lower to have a smooth response.
  - **viscosity**: viscosity coefficient to compute basic friction forces.
  - **resolution**: resolution max of the textures rendered in the GPU. 128 or 256 are considered as high values, and will give a very precise detection. But 32 or 64 give the best trade-off.
  - **resolutionPixel**: If this parameter is strictly positive, it will be used as the resolution in millimeter of a pixel in the collision detection. Each pixel of the texture will correspond to a square of $resolutionPixel * resolutionPixel$ in the scene world. Like this, every interactions will have the same precision. If this parameter is not used, every interaction between two objects will be rendered with a given resolution. It means that a high precision will be used at the beginning of the interaction. When the smaller the intersubsection AABB of a pair of TriangleModel , the higher the precision.

- **adaptiveResolution**: If this parameter is strictly positive, then adaptive resolution will be used: far collisions will be treated with a lower resolution than collisions occurring right in front of the camera. The value is the distance to switch from a level of resolution to another.
- **depthBB**: Depth of the hierarchical AABB computed for every TriangleModel. They are used during the BroadPhase, to eliminates pairs of non-colliding objects. The fewer pairs returned by the BroadPhase, the better for the performances, as less DepthPeeling (the real bottleneck of the algorithm) will be done.
- **GPUCollisionVolume**: Activate Cuda algorithms. The NarrowPhase will be performed fully on the GPU using a Cuda implementation of the NarrowPhase.
- **globalRendering**: do only ONE DepthPeeling of the resulting AABB of all the AABB in collision. The resolution used is still "resolution"
- **useGlobalBBox**: You can reduce the space where the collision will be computed to a global bounding box. If you want to compute the collision only into "globalBBox", you need to activate "useGlobalBBox".
- **globalBBox**: the global Bounding Box where the collision will be computed. Only used if "useGlobalBBox" is active.
- **verbose**: print lots of information about the state of the collision pipeline.

- **LDIPenalityContactForceField** : When a collision is detected between two TriangleModels, a LDIPenalityContactForceField will appear in your scene graph, performing the per-pixel contact forces computed and accumulated at the vertices. Contrary to the default Sofa implementation, no mapping to temporary dofs is necessary. We only create contacts and an interaction force field.
- **RelaxtionSolver** : experimental

1.2 Collision Pipeline

The Sofa documentation (http://www.sofa-framework.org/sofa/doc/sofadocumentation.pdf) will give an overview of the collision pipeline in Sofa. We just kept the same scheme. All these methods are implemented in LDIDetection.h . Basically, we try to identify the colliding objects, and their interactions using two passes.

- **BroadPhase**: This pass will quickly eliminate most of the possible colliding pair of elements, keeping only a few for which we will need an extended pass to determine if the pair is really in collision. This is done by a basic AABB intersubsection.
- **NarrowPhase**: Using the remaining pairs of elements, we perform a DepthPeeling of these two elements, in the intersecting AABB in the 3 directions. Then, we analyse the 3D textures obtained and detect eventual collision by analysing the normals of the triangles. More details are given in the paper. We accumulate in a vector all the collision found, storing the barycentric coefficient in the two colliding triangles, the two triangles themselves, and the surface of the interaction. We deal with textures, not with triangles directly.

1.3 Collision Response

Instead of creating new Dofs at the exact location of the collision and linking them to the collision models using mappings, and applying to them interaction forces, we process all the contact pixels once, and accumulate their contributions to the vertices of the triangles of the models. The collision pipeline calls the contact manager to create the adequate response forces. LDIPenalityContact.h implements this part of the pipeline. In function SetDetectionOutputs, the vector previously filled with the interaction pairs is analysed. Some redundant computations and the whole volume of intersubsection are then computed, and then the Interaction Forces are set, using the LDIPenalityContactForceField associated to the LDIPenalityContact.
In LDIPenalityContactForceField.h, we define the method **addForce** and **addDForce** using the same equations as described in the paper[**?**].

2 Installation

This Collision Detection and Response method is distributed with the QPL licence. This package works with the Sofa Framework 1.0 beta 3 freely available at : http://www.sofa-framework.org/download

SOFA is an Open Source framework primarily targeted at real-time simulation, with an emphasis on medical simulation. It is mostly intended for the research community to help develop newer algorithms, but can also be used as an efficient prototyping tool.

You must have the GLEW library (The OpenGL Extension Wrangler Library) installed in your system. You can download it freely at http://glew.sourceforge.net/.

2.0.1 Hardware

The implementation intensively uses the graphics hardware, using the Depth Peeling algorithm. Vertex and Pixel shaders must be available. The minimum required to run it should be GeForce6 series(or equivalent), but we recommend GeForce8 series. We provide in this package a tool to process Depth Peeling (DepthPeelingUtility) on arbitrary triangular volume objects. Given one, or several Triangle models, it will fill a 3D texture containing the depth layers along a given orientation. An application of this method could be a voxelizer.

2.0.2 Compilation

See README file.