C

Low level implementation details

As discussed in Chapter 4 the 3D reconstruction code is based on the video-based rendering algorithm as presented by Li et al. (2003a). I implemented this algorithm using C++ and OpenGL code that runs under Linux and Mac OS X. I developed a HandOfGod object that encapsulates all of the processing used in Li et al.'s VBR algorithm. The object is able to be used by other scene graph-based rendering applications, such as Tinmith, to generate a HOG object that exists as a node in the scene graph. Therefore the object can be transformed with translation, scaling and rotation operations. The object also handles processing of data received from the HOG table. The object has the following two methods (see Figure C.1):

1. **processPacket()** - This method is passed packets from the instantiating program. The packets expected are either contour or image data. The data is processed and stored in a convenient format ready to be rendered. The internal storage differs depending on the network used, this is discussed further in section 4.2.1 and 4.2.2. This method is not responsible for reading data off of the network, this is managed by the lower level networking code of the instantiating program.

2. **render()** - This method makes the OpenGL calls necessary to perform the 3D reconstruction as described by Li et al's VBR algorithm. It loads the appropriate textures into texture memory on the graphics card, draws the geometry and then applies the textures.

The HandOfGod object is instantiated by the visualisation system, either a VR system or Tinmith (see Section 4.3). Networking is handled by the visualisation

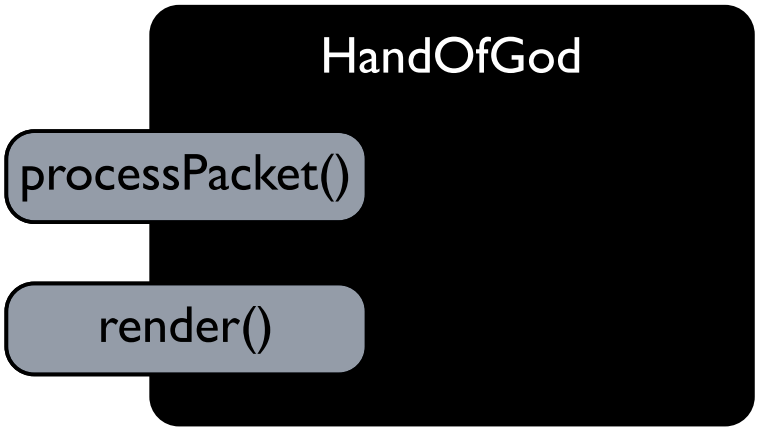

Figure C.1: Black box representation of the HandOfGod object.

system and only packets relating to the HOG objects are forwarded to the packet handler of the HandOfGod object. The HandOfGod object is rendered as a node of the visualisation system's scene graph.