Practical – 3D model based tracking

This practical aims to track an object, in images acquired by a camera, and to simultaneously estimate its pose, knowing a 3D model of it. The tools that are going to be used are a digital camera, for the hardware, and the C++ language, CMake and ViSP library for the software side. ViSP stands for Visual Servoing Plaform and comes with its documentation. This is a framework for vision-based control of a robot and visual tracking using several features. CMake is a C++ project configuration tool to be used with many IDEs.

To configure the practical programming project:
Run CMake, select the practical source and build directories and then configure and generate.

The wireframe model based tracking follows the same ideas as the point-based one. However, some visibility tests on the object model, the error and feature computation and image processing are a bit more complex. That is why the ViSP library encapsulates all the low level image processing and pose optimization in the single method *track* of a *vpMbEdgeTracker* object. To be usable, a set of files needs to be created and loaded by the tracker object.

1. Use the *loadConfigFile* method to load an *xml* file named *box.xml* where the camera parameters and the tracking option are set. Adapt the intrinsic parameters to your camera.
2. Use the *loadModel* method to load a *cao* file named *box.cao* where the 3D model of the box is defines. Adapt the vertices coordinates to the size of your box.
3. Extract the camera parameters from the tracker object to a *vpCameraParameters* object using the *getCameraParameters* method.
4. Initialize the object pose by using the *initClick* method of the tracker. It takes among its parameters a *box* file, without extension. It loads the *box.init* file, including the coordinates of four particular vertices as visible in the *box.ppm*, an image displaying a similar box.
5. Then, get the initial pose using the *getPose* method and display the model using the *display* method of the tracker object.
6. Use the *track* method to optimize the pose on contour image measurements and again, in the image acquisition loop track the object along images.
7. Display the model in the loop and use the static *vpDisplay::displayFrame* to display the object frame in the image. Set the display of moving edge elements using the *setDisplayFeatures(true)*.