VIAM-GenoM

David Marquez-Gamez
Michel Devy
LAAS - CNRS

May 16, 2011

Abstract

The Versatile Image Acquisition Module (viam) handles image acquisition from firewire DCAM cameras through the libdc1394 and OpenCV libraries or USB cameras through the video4linux API. It can also read images from files, either saved from a previous acquisition or not. It is open-source and released under a BSD license. A semi-automatized camera calibration procedure usign OpenCV algorithms is included. Simple filters can be applied to the raw images, like distortion correction or image rectification for stereo pairs. Gain, exposure, white balance, debayerization and other hardware parameters can also be controlled, either by software or directly by the camera if the hardware enables it.

1 Overview

This document is divided into XXX parts. The first one describes the installation of Robotics Package Collection (robotpkg). The second part explains how to install the VIAM-GenoM module using robotpkg so it can be easily built without knowing about the package's building details. The third part .....

2 Terminology

Here is a description of all the terminology used within this document.

Package A set of files and building instructions that describe what's necessary to build a certain piece of software using robotpkg. Packages are traditionally stored under `/usr/local/openrobots/src/robotpkg/`.

robotpkg This is the The Robotics Package Collection. It handles building (compiling), installing, and removing of packages.

Distfile This term describes the file or files that are provided by the author of the piece of software to distribute his work. All the changes necessary to build are reflected in the corresponding package. Usually the distfile is in the form of a compressed tar-archive, but other types are possible.

Precompiled/binary package A set of binaries built with robotpkg from a distfile and stuffed together in a single `.tgz` file so it can be installed on machines of the same machine architecture without the need to recompile.

Sometimes, this is referred to by the term "package" too, especially in the context of precompiled packages.

Program The piece of software to be installed which will be constructed from all the files in the distfile by the actions defined in the corresponding package.

3 Typography

When giving examples for commands, shell prompts are used to show if the command should/can be issued as root, or if "normal" user privileges are sufficient. We use a `#` for root's shell prompt, and a `%` for users' shell prompt, assuming they use the C-shell or tcsh.

4 Installing robotpkg

4.1 Getting robotpkg

Decide in which directory you'll install robotpkg. We'll assume `/usr/local/openrobots` in this document. The `/usr/local/openrobots` is a mount point for the shared openrobots software.

- As your regular user, simply run in a shell:

  ```
  % mkdir -p /usr/local/openrobots
  ```

  If the above command fails due to insufficient permissions, do the following:

  ```
  % sudo mkdir -m 755 -p /usr/local/openrobots
  % sudo chown `id -u` /usr/local/openrobots
  ```

- Clone the git repository for robotpkg to your work space:

  ```
  % cd /usr/local/openrobots
  % mkdir src
  % cd src
  % git clone ssh://softs.laas.fr/git/robots/robotpkg
  ```

  If the above command fails due to insufficient permissions, run the following:

  ```
  % git clone http://softs.laas.fr/git/robots/robotpkg.git
  ```

4.2 Bootstrapping

- Edit your shell config file in order to update your PATH, LD_LIBRARY_PATH and PKG_CONFIG_PATH environment variables:

  - If you're using [t]csh this can be done by adding the following to $HOME/.tcshrc:

    ```
    # OpenRobots
    setenv ROBOTPKG_BASE /usr/local/openrobots
    setenv PATH ${ROBOTPKG_BASE}/bin:${ROBOTPKG_BASE}/sbin:${PATH}
    setenv LD_LIBRARY_PATH ${ROBOTPKG_BASE}/lib:${ROBOTPKG_BASE}/lib/openp
    setenv PKG_CONFIG_PATH ${ROBOTPKG_BASE}/lib/pkgconfig:${PKG_CONFIG_PAT
    ```

  - If you're using bash or ksh or zsh, add the following to $HOME/.bashrc or $HOME/.bashrc or $HOME/.zshenv:

```
# OpenRobots
export ROBOTPKG_BASE=/usr/local/openrobots
export PATH=${ROBOTPKG_BASE}/bin:${ROBOTPKG_BASE}/sbin:${PATH}
export LD_LIBRARY_PATH="${ROBOTPKG_BASE}/lib:${ROBOTPKG_BASE}/lib/open
export PKG_CONFIG_PATH="${ROBOTPKG_BASE}/lib/pkgconfig:${PKG_CONFIG_PA
```

  - Relaunch your shell in order to take into account the changes

- Run the bootstrap script from /usr/local/openrobots/src/robotpkg/bootstrap/

  - If you have already installed openrobots and want to overinstall then first remove your configuration file:

    ```
    % rm /usr/local/openrobots/etc/robotpkg.conf
    ```

  - else simply run the bootstrap script:

    ```
    % cd /usr/local/openrobots/src/robotpkg/bootstrap
    % ./bootstrap --prefix=/usr/local/openrobots
    ```

5 Using robotpkg to install VIAM-GenoM

Go to the sub-directory of robotpkg corresponding to the viam-genom package and run make update.
make update installs the package and all the other packages that could be affected by the rebuild of this package.

```
% cd /usr/local/openrobots/src/robotpkg/image/viam-genom
% make update
```

6 Using the VIAM Module

6.1 General procedure

- Launch h2 init to initialize communication libraries

  As your regular user, simply run in a shell:

  ```
  % h2 init
  ```

If the above command fails, run the following:

```
% /usr/local/openrobots/bin/h2 init
```

- Start a server `tclserv` on the same shell

  ```
  % tclserv
  ```

  If the above command fails, run the following:

  ```
  % /usr/local/openrobots/bin/tclserv
  ```

- In another shell, start the VIAM module

  ```
  % viam
  ```

  If the above command fails, run the following:

  ```
  % /usr/local/openrobots/bin/viam
  ```

- In another shell, start a tcl shell with the package genom

  ```
  % elwish -package genom
  ```

  If the above command fails, run the following:

  ```
  % /usr/local/openrobots/bin/elwish -package genom
  ```

- From tcl (eltclsh) shell, connect to the machine running `tclserv`

  ```
  eltclsh> connect
  ```

- From tcl (eltclsh) shell, load the functions to communicate with the viam module

  ```
  eltclsh> lm viam
  ```

- From tcl (eltclsh) shell, send the requests to the viam module, so the module is running and ready to requests.

6.2 Display camera view

From tcl (eltclsh) shell, send the requests to the viam module.

For example with iBot camera:

```
eltclsh >::viam::CameraCreate iBot dc1394:0x00d0f54000000179
eltclsh >::viam::BankCreate iBotBank VIAM_DOUBLE_BUFFERING VIAM_DISABLE
eltclsh >::viam::BankAddCamera iBotBank iBot iBotImage
eltclsh >::viam::CameraSetHWMode iBot VIAM_HWSZ_640x480 VIAM_HWFMT_YUV422 VIAM_
eltclsh >::viam::Init
eltclsh >::viam::Configure iBotBank
eltclsh >::viam::Acquire -ack iBotBank 0
eltclsh >::viam::Display iBotBank iBotImage VIAM_ON VIAM_OFF 0 0
```

7 Killing the module

From tcl (eltclsh) shell

```
eltclsh> killmodule viam
```

8 Cleaning everything

Run in a shell

```
% h2 end
% pkill tclserv
```