**Keysens**

# *RuntimeGigE* User's Manual

Document title: *RuntimeGigE* User's Manual
Program version: 1.4
Document number: 2002
Document revision: 01
Document Date: June 2017

# Contents

# 1   Introduction

***Keysens RuntimeGigE*** is a runtime program developed by ***Keysens*** to execute vision projects in real time. Vision projects are designed with the configuration program ***vDevelop***. They implement machine vision applications using ***Keysens*** algorithms script approach. ***RuntimeGigE*** runs on ***Microsoft Windows*** Operating Systems such as ***Windows 7***, ***Windows 8*** and ***Windows 10***. Since it is a runtime program intended for industrial applications, it is recommended to run it on an embedded device under an embedded operating system, such as ***Microsoft Windows Embedded Standard 7***.

***RuntimeGigE*** comes installed in some ***Keysens Vision Processor*** models, particularly those intended to control ***GigE Vision*** cameras. ***Keysens Vision Processors*** are embedded devices used to run real industrial machine vision applications.

The purpose of this manual is to explain the usage of the program. Figure 1 shows a screen shot of the program in a real situation.

For more technical information on setting up machine vision applications please see ***Keysens*** technical notes. Particularly "Telegrams technical note", "Configuration files technical note" and "Project files technical note".
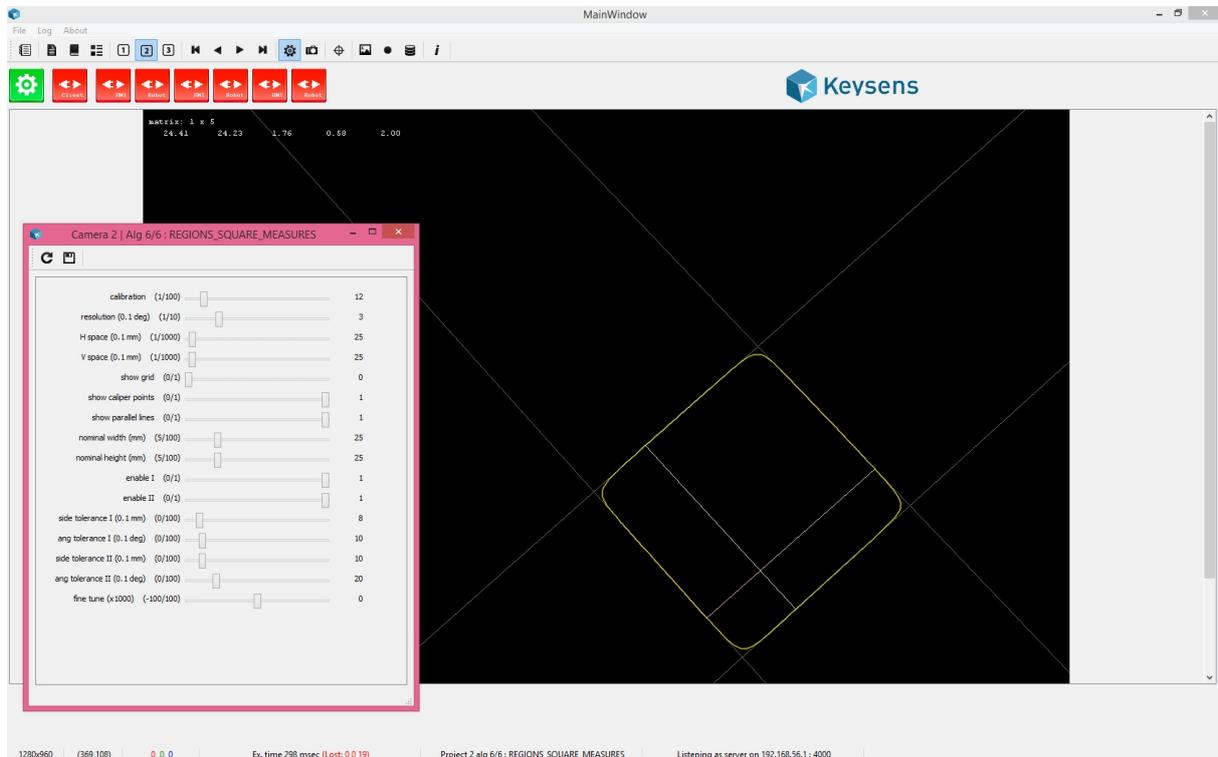
Figure 1: A screen shot of *RuntimeGigE* executing a project

## 2   Program installation folders

The objective of *RuntimeGigE* is to execute vision projects in real time.  To do so, the program needs to access collections of vision algorithms and the projects to execute, which consist of algorithms scripts.  *Vision Processors* have vision algorithms libraries installed and other resources needed by the program. These files are placed in the following folders (`.` means the root installation folder, usually `/runtimeGigE`):

| Folder | Contains |
|--------|----------|
| `./alg` | Algorithms libraries and algorithms description files. |
| `./release` | The program executable and runtime libraries. |
| `./cfg` | Configuration files, vision projects, model images and model data. |

## 3   Program window areas

*RuntimeGigE* opens one main window and some secondary windows for performing punctual operations or for showing log messages: a *log window*, a *camera parameters window* and an *algorithm parameters window*.

The log window can be made visible and invisible by the user. When not shown, it is still logging program messages.

The camera parameters window appears when the user opens it to modify the parameters of the selected camera.  Up to three cameras can be connected.  The camera parameters

window relates to the selected camera when opening the window.

The algorithm parameters window appears when the user opens it to modify the parameters of the selected algorithm of the vision project assigned to the selected camera.

The main window is always visible and is divided into several areas for displaying different information. The areas and the information shown are:

| Area | Shows |
|---|---|
| Menu bar | Menu names. |
| Tool bar | Tool bar buttons. |
| Results image | Images and data results produced by the selected algorithm. |
| Connetions area | Icons showing the work mode and the connected devices: configuration client, robots and HMIs. |
| Logos area | Developer logo and application logo. |
| Status texts | Several information displayed: image pixel values, execution times, lost frames, selected camera and selected algorithm, IP/port the program is listening as server for configuration. |

The *Keysens* developer logo appears on the right side and is fixed. The application logo appears in the centre, it is an icon loaded by the program on start up. The file is `./cfg/logo.png`.

# 4 Work mode

*RuntimeGigE* has two work modes, *run* and *stadby*.

In *run* mode, the normal mode, the program is continuously acquiring images from the cameras, applying the algorithms scripts of the vision projects and sending results to the connected devices. In this mode, the icon showing a cogwheel appears green.

When a configuration client (a PC running the configuration program *vDevelop*) connects to *RuntimeGigE*, or when a device in the installation sends the telegram [STB], the program enters in *standby* mode. Images are acquired but no algorithm is applied. This mode is entered because the configuration client will usually change a vision project, or because a device in the installation orders the program to enter this mode. The icon showing a cogwheel appears red.

To enter the *run* mode again, both following events have to occur:

- a telegram [RUN] has to be received from any connected device to cancel a previous telegram [STB],

- and the configuration client has to be disconnected.

# 5 Connected devices

A configuration client can connect by TCP/IP protocol to upload and download vision projects. When connected, the icon showing the legend *client* appears green. When not connected, it appears red.

Besides, six devices in the installation can connect to the program also by TCP/IP, two for each connected camera. They are supposed to be a robot and an HMI, but can be any

TCP/IP device acting as client or as server. The program will send the results of the each project to the corresponding connected devices.

*RuntimeGigE* can behave as a server or as a client with these six devices. The selection is made in the configuration file `./cfg/port.cfg`. The first row of the file is the port number for the configuration client. the second row indicates if *RuntimeGigE* is server or client for connetions with robots and HMIs. If the second row contains a `1` the program creates six TCP/IP servers for these connexions, if it contains a `0` (or something different from a `1`), it creates six TCP/IP clients.

When a device is connected, its icon appears green. When it is not connected, it appears red. From left to right, the first two icons showing legends *robot* and *HMI* correspond to the first camera, the following two icons correspond to the second camera and the last two icons correspond to the third one.

# 6    Vision projects

*RuntimeGigE* executes up to three vision projects at the same time, each one assigned to a connected camera (up to three). A configuration client can upload and change any project. Besides, connected devices in the instalation (robots and HMIs) can also change any project sending a `[PRO999]` telegram.

For more information on projects and telegram see "Telegrams technical note" and "Configuration files technical note".

# 7    Menus

*RuntimeGigE* is a runtime program intended to be left alone running on an embedded device. So it does not have many menu options. The user performs actions mainly by the tool bar buttons.

The program has these menus and options:

| Menu/option | Action |
|---|---|
| File/Exit | Exits program. |
| Log/Show hide log | Shows and hides the log window. |
| Log/Print project | Prints the vision project of the selected camera. |
| Log/Print camera features | Prints the selected camera features. |
| About/Version | Opens a message box showing the program version. |

# 8    Selecting a camera

Almost all actions performed by the user when interacting with the program refer to the selected camera. There can be up to three cameras connected to a *Vision Processor* and controlled by *RuntimeGigE*. A camera is selected with these tool bar buttons:

| Tool | Action |
|------|--------|
| [1] | Selects camera one. |
| [2] | Selects camera two. |
| [3] | Selects camera three. |

# 9  Saving images and image sequences

Saving captured images is a very important tool for developing machine vision applications. When objects in motion pass below the cameras, you can acquire image sequences to see how objects are detected, to see if images are clear and sharp despite of the objects motion, if light is good enough, etc. Then you can use the images to develop and to test a vision project with the configuration program *vDevelop*.

These tool bar buttons are used to save images captured from the selected camera.

| Tool | Action |
|------|--------|
| 🖼 | Keeps in memory the next image captured with the selected camera. |
| ● | Keeps in memory the next sequence of 15 images taken with the selected camera. |
| 🗄 | Saves the last image or image sequence captured with the previous buttons. |

# 10  Selecting algorithms results

The algorithms results of the selected camera can be selected with four tool bar buttons:

| Tool | Action |
|------|--------|
| ⏮ | Selects the first algorithm or what is the same, the original image. |
| ◀ | Selects the previous algorithm. |
| ▶ | Selects the next algorithm. |
| ⏭ | Selects the last algorithm. |

# 11  View options

*RuntimeGigE* shows numeric data results as over-impressed text in the image area of its main window. This text can be drawn in different colours for making it more visible. There are some predefined colours. The colour can be changed by a context menu over the image area (opened with the right mouse button).

| Image contextual menu option | Action |
|---|---|
| Change font colour | Changes the data draw colour to some predefined colours. |

Besides, the program can draw two red lines, a vertical and a horizontal one, to point out the centre of the image. The lines can be made visible and invisible with a tool bar button.

| Tool | Action |
|---|---|
| ⊕ | Makes visible and invisible the lines showing the centre of the image. |

## 12   Printing projects, features and parameters

Three tool bar buttons print information on the log window, one prints the project of the selected camera and another two print camera features.

| Tool | Action |
|---|---|
| 📄 | Prints the project of the selected camera in the log window. |
| ▆ | Prints the most important features of the selected camera in the log window. |
| ▤ | Prints all features of the selected camera in the log window. |
| ✦ | Prints persistent parameters in the log window. |

## 13   Showing and hiding auxiliary windows

*RuntimeGigE* informs of events in a log window. This window is usually not visible, but the user can make it visible and invisible with a tool bar button. The contents of the log window can be saved and cleared with menu options in the log window.

Besides, the user can open two more windows to change parameters of the selected camera and to change parameters of the selected algorithm of the selected camera. These two windows have two tool bar buttons, one to recover the values from the project loaded in memory and one to make the values permanent in the project, the program then saves the project to disk.

| Tool | Action |
|---|---|
| 📋 | Opens and closes the log window. |
| 📷 | Opens and closes the camera parameters window. |
| ⚙ | Opens and closes the algorithm parameters window. |

# 14    Measuring the vision projects execution time

*RuntimeGigE* measures the time it takes to execute the algorithms script for every project. Measurements are printed in the status bar of the main window. The time is measured in milliseconds. The time shown corresponds to the execution of all the algorithms of the project assigned to the selected camera.

Important: The time to process a vision project has to be smaller than the frame rate period of that camera, so that no image is lost or not processed. The status bar shows also, in parenthesis, how many images have been lost for each camera. For a vision application to be correct, no image has to be lost. In case of a loss, *RuntimeGigE* does not send any result to the robot or HMI connected to that camera, since the results would be available out of time and that may cause problems when a robot is tracking objects. In resume, this situation has never to happen.

# 15    On how algorithms are applied

*RuntimeGigE* disposes of three execution threads to apply each of the three projects to the images captured by the three connected cameras. If a camera is not connected, no project is executed for that camera.

The algorithms of each project are applied sequentially following the project algoritms script. If the last algorithm produces numeric data, messages with these results are sent by TCP/IP to the connected devices (robot and HMI) assigned to that camera.

# 16    Comments

If you experience any problems with this document or want to give us feedback, please email us at info@keysens.com.