User Manual

Project Kitchen Occupation

TSBB11 HT 2013

Version 1.0

Status

Reviewed	–	2013–12–13
Approved		

2013–12–13

Project Kitchen Occupation

Bilder och Grafik CDIO, HT 2013
Department of Electrical Engineering (ISY), Linköping University

Participants

Name	Tag	Responsibilities	Phone	E-mail
Mattias Tiger	MT	Project manager	073–695 71 53	matti166@student.liu.se
Erik Fall	EF	–	076–186 98 84	erifa226@student.liu.se
Gustav Häger	GH	System integration	070–649 03 97	gusha124@student.liu.se
Malin Rudin	MR	–	073–800 35 77	malru103@student.liu.se
Alexander Sjöholm	AS	–	076–225 11 74	alesj050@student.liu.se
Martin Svensson	MS	Documentation	070–289 01 49	marsv106@student.liu.se
Nikolaus West	NW	Testing	073–698 92 60	nikwe491@student.liu.se

Homepage: TBA

Customer: Joakim Nejdeby, Linköping University, Origo 3154
Customer contact: 013–28 17 57, joakim.nejdeby@liu.se
Project supervisor: Fahad Khan, Linköping University, fahad.khan@liu.se
Examiner: Michael Felsberg, michael.felsberg@liu.se

Contents

List of Figures

List of Tables

Document history

Version	Date	Changes	Sign	Reviewed
0.1	2013–12–13	Initial draft	MS	MT

1 Installing the system

1.1 Hardware

Each Kinect camera must be installed above a door with no overlapping view shared with any other Kinect camera. The Kinect must point down or slightly angled towards the room. For optimal results the Kinect should be placed approximately 40 cm above the door to be able to detect even the tallest persons. Each Kinect must be connected to a power source, and to a device running the system software using USB.

1.2 Software

There are two versions of the software, one with a calibration and configuration GUI and one lightweight version without GUI. In order for the lightweight version to work two configuration files need to be located next to the executable. The files are:

- mainConfig.yml
- masks.yml

When running the GUI version there is exists an additional configuration file specifically for the GUI:

- guiConfig.yml

Default versions of these files are located in the conf folder. When the GUI version is run these files are generated and stored next to the executable. If desired these can be copy to the conf folder and act as defaults in the future.

Linux, OS X or Windows is required on the machine running the software. At least one Kinect camera must be connected before starting the program. Some software libraries are required to compile the program, these are listed in table 1.1 below.

Software	**Comments**
OpenCV2	Needed for general image processing
libFreenect	Needed for communication with Kinect on Linux and OS X systems
OpenNI	Needed for communication with Kinect on Windows systems
libCurl	Needed to send HTTP requests to the report API
QT5	Needed for the GUI code, not used in headless variant

Table 1.1: Required software libraries.

2 Setting up the system

The easiest way to setup the system is by using the GUI. Here the main settings can be adjusted. However, there are some advanced settings that can only be adjusted in the configuration file. More on this later.

2.1 Calibration

The system needs to be calibrated for the current installation height of the Kinect sensor. This is easily done in the Calibration Window which can be started under the System tab in the menu bar. To the left is a thresholded depth image and to the right a histogram. The slider below the histogram adjusts the threshold level. The threshold should be set so that a "normal" person's chest is not removed by the thresholding. Press Apply to save the changes.

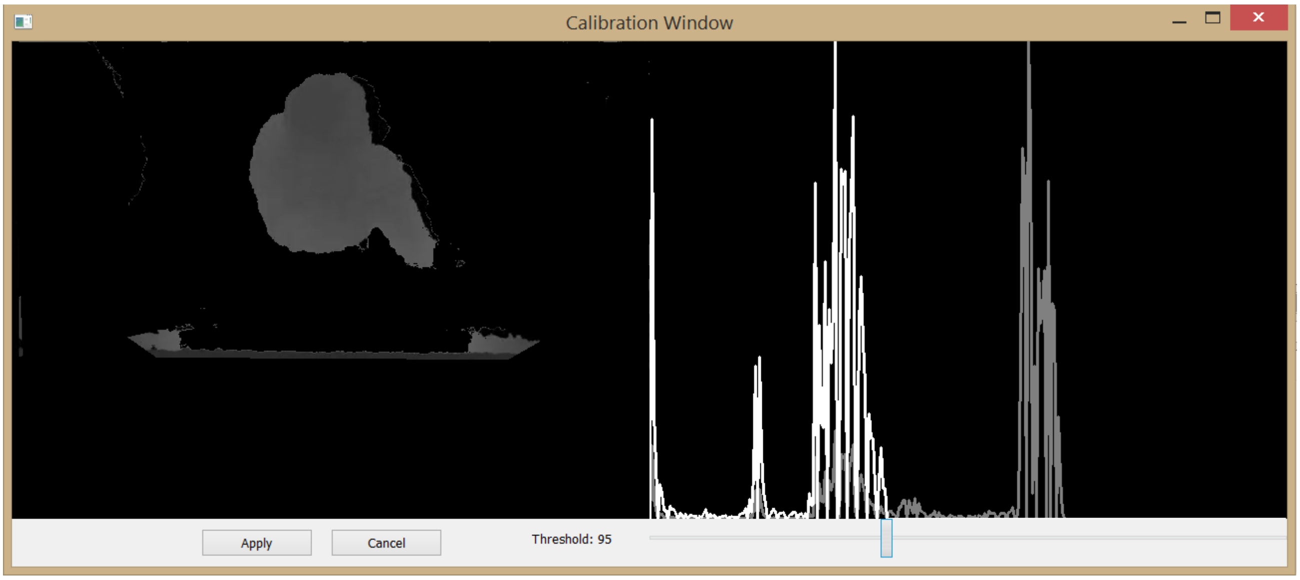

Figure 2.1: *Calibrating the depth threshold of the system. A histogram is shown to help the user. The adjusted histogram is shown in white and the raw histogram in gray.*

2.2 Configuration

Before the system can run properly some masks needs to be defined. These are used to:

- Define doors
- Define non-interesting areas
- Define entering checkpoints

These are adjusted in the Configure Window found under the System tab in the menu bar.

2.2.1 Checkpoint circles

The first thing to do is to insert three circles that are used as checkpoints when the system is counting people. All three circles needs to be passed before the system accepts it as and enter or exit. To draw a circle, check the circle check box. Place the cursor where want the center of your circle to be. Now press the mouse and move the cursor to adjust the radius. Three circles are automatically drawn and they should cover the door as shown in figure 2.2 below.

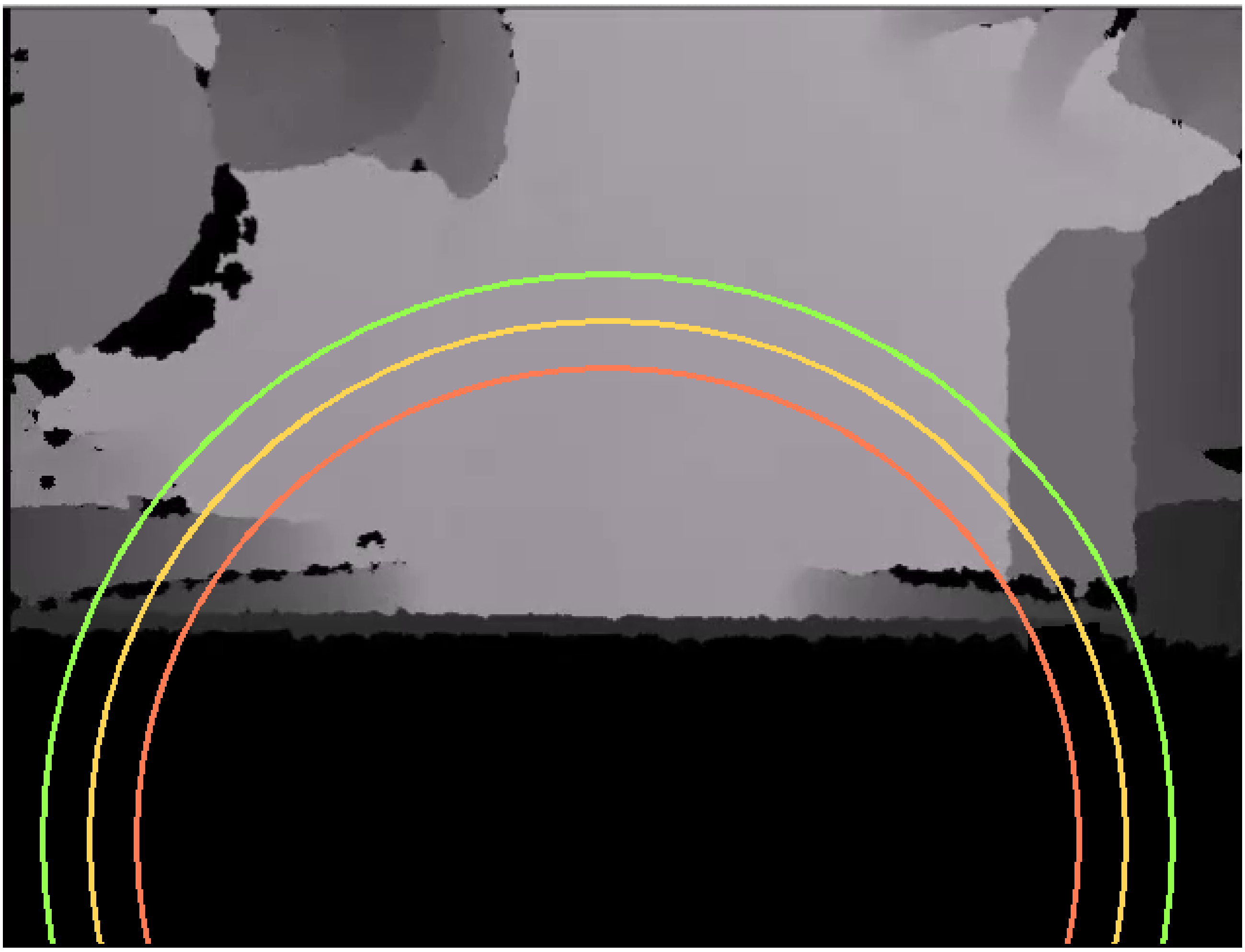

Figure 2.2: *A preferred placement of the checkpoint circles.*

2.2.2 Door mask

The door mask should cover the entire door with some surrounding in front of the door. It is better to make this area too big rather than too small since people entering the room need to appear inside of this mask in order to be counted. A good example is shown in figure 2.3 below.

To add a door mask press *New Polygon*. Now draw the mask by clicking in the image which adds control points to the mask, forming a polygon. To undo press Ctrl + Z. When the mask is done press *Add as door*.

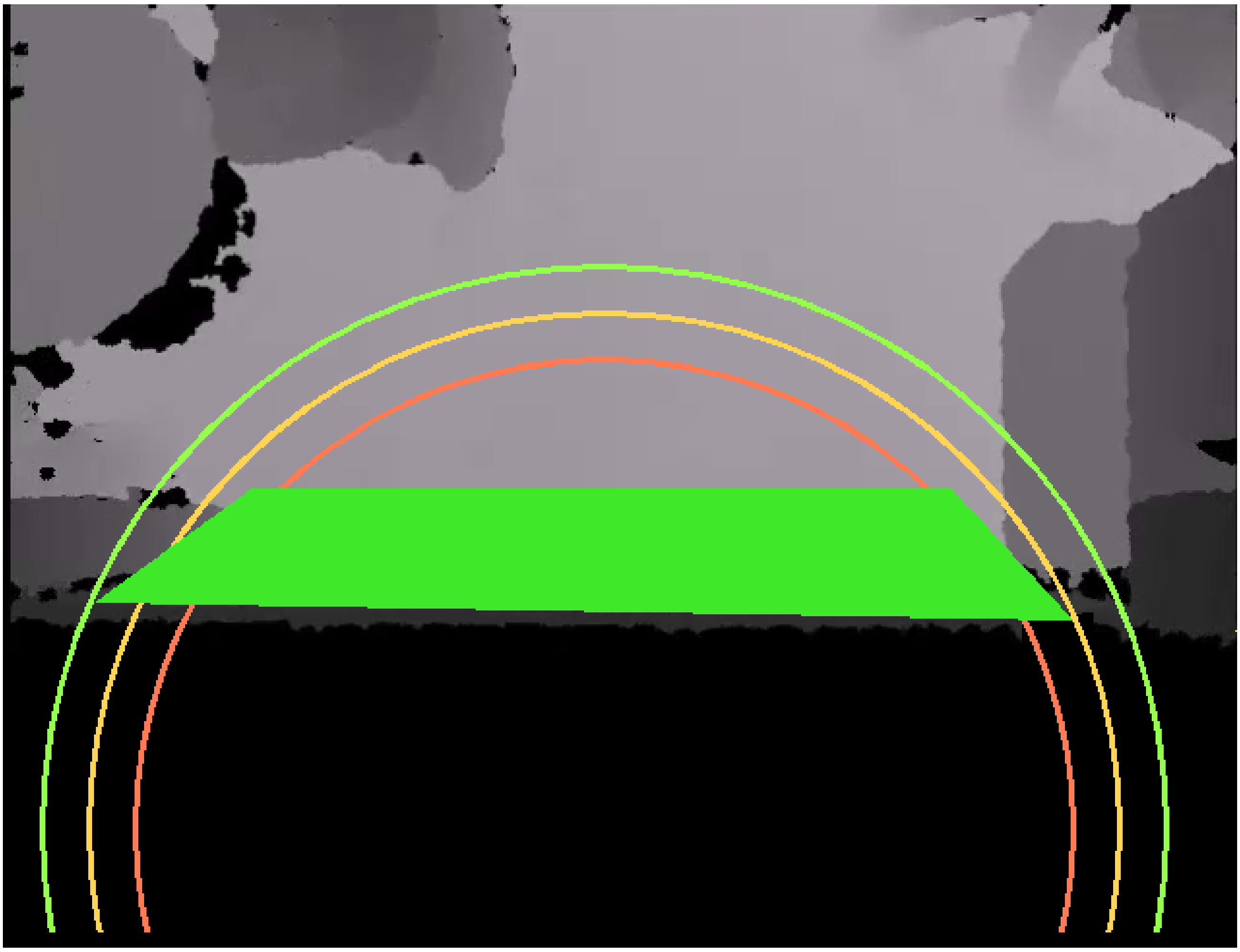

Figure 2.3: *The preferred placement of the door mask is showed in green.*

2.2.3 Exclusion mask

Exclusion masks should cover areas that should be completely ignored by system e.g. where people can not walk or appear. This can be areas like tables or areas above the door (walls in this case). This is illustrated in figure 2.4 below. Note that for long usage of the system, movable furniture should not be excluded.

To add an exclusion mask press *New Polygon.* Now draw the mask by clicking in the image which adds control points to the mask, forming a polygon. To undo press Ctrl + Z. When the mask is done press *Add as exclusion.*

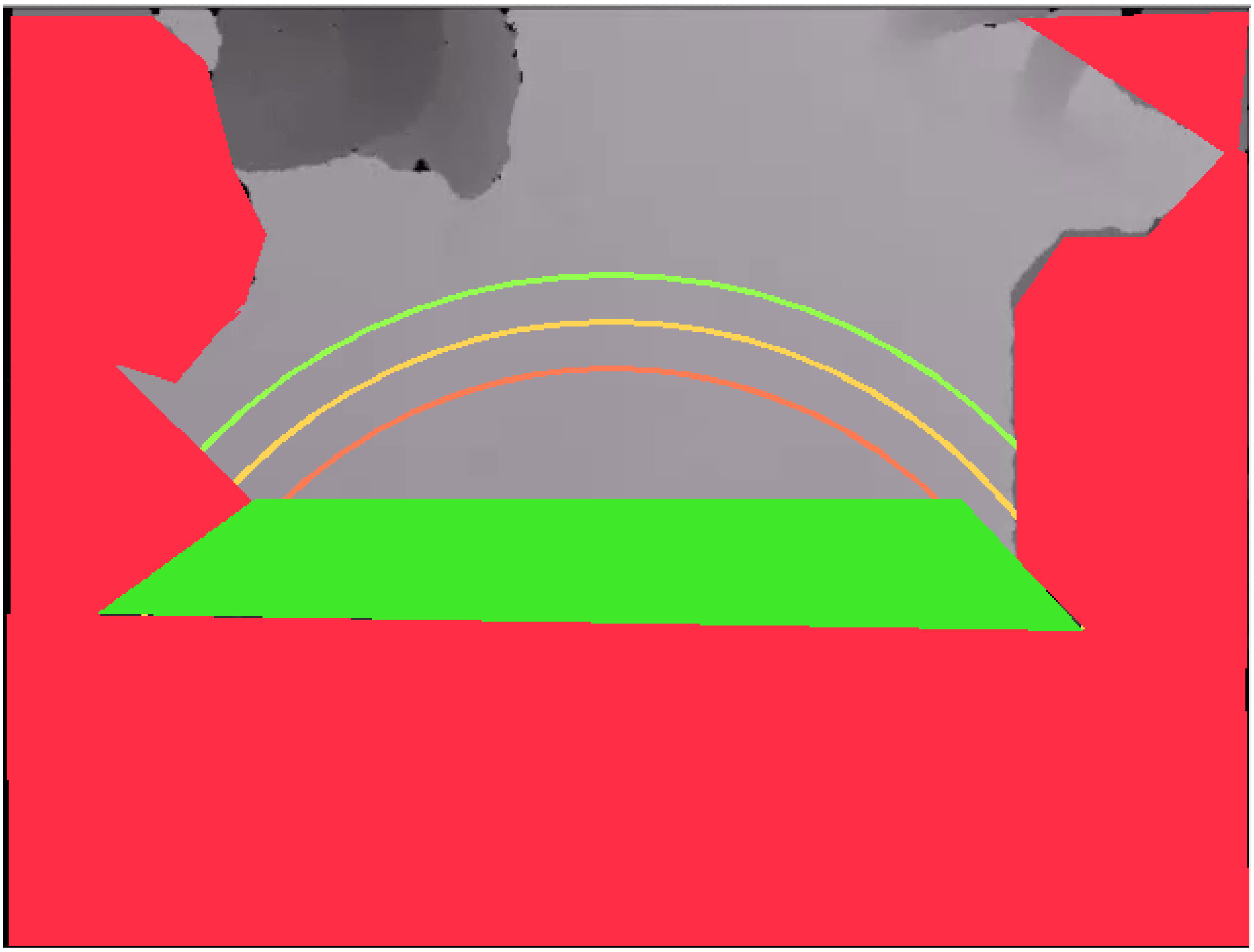

Figure 2.4: *Exclusion mask is marked as red. It covers areas where people can not walk or appear.*

2.3 Configuration file

The config file used by the algorithm pipeline is called *mainConf.yml*. In it any configurable variable in any algorithm currently selected to be in the pipeline can be specified. The pipeline itself can also be specified, allowing fo rapid swapping of algorithms. The most useful variables in the current pipeline are shown in table 2.1.

Variable	**Algorithm/Program-part**	**Description**
runFromFile	Network	If set to 1, the video sources are files found in the paths *videoFilePaths*.
videoFilePaths	Network	The paths to the video files used if running from file.
useKinect	Network	If set to 1, the video sources are from Microsoft Kinect cameras.
TrackingMaximumDistance	Tracking	The maximum distance an object can be considered to have moved since last frame.
TrackingMinimzumLifeSpan	Tracking	The minimal time (in # frames) a potential object must have existed (and been tracked) before it is considered a real object.
TrackingMaximumTimeLost	Tracking	The maximum time (in # frames) an object is allowed to be lost before it is forgotten.
lowestDistanceOverFloor	Kinect Segmentation	The limit (height units) of how short a person can be. Set this variable using the GUI calibration utility described previously.
webServerUrl	Network	The address to the web service to which results are reported.
lowOccupancyThreshold	Queue Severity Estimation	If there are more people than this and less than highOccupancyThreshold, queue severity will be set to 1 (medium). Setting these value is optional but recommended for rooms where an overwhelming proportion of the occupants will be in a queue (e.g. rooms without places for sitting). If no values are set the severity is solely determined by visible queues.
highOccupancyThreshold	Queue Severity Estimation	If there are more people than this, queue severity is 2 (high). This value must be higher than lowOccupancyThreshold if set.

Table 2.1: The most useful and common variables in the current pipeline.

Currently the pipeline consists of two major algorithms: *ImageProcessor* and *Analytics*. These in turn have several sub-algorithms that are executed in the order specified in the config file. The current pipeline is structured in the following way:

ImageProcessor:
- *KinectSegmentation*
- *TrackingBruteForce*

Analytics:
- *EntryExitCounter*
- *FlowEstimator*
- *QueDetector*
- *QueSeverityEstimator*

Any algorithm registed in the system can be used as a subalgorithm for any other algorithm, writing in the config file in the same way as with *KinectSegmentation* being a sub algorithm to *ImageProcessor*. To get an empty algorithm placeholder any none-registerd algorithm name (or variable name) may be used, such as:

UnregisterdName:
- *KinectSegmentation*
- *...*

It can now be used as a sub-algorithm to another algorithm (or placeholder algorithm):

ImageProcessors:
- *UnregisterdName*
- *...*

A placeholder algorithm works by just passing through initialization and processing calls to its sub-algorithms.

Warning: If you do not know what your are doing, do not modify the algorithm pipeline. Some algorithms have requirements which must be provided by earlier algorithms, the system will not run if these are not met. See the code documentation for further details on requirements and effetcs of different algorithms.

2.4 Debugging from GUI

When setting up and especially when developing the system it is convenient to use the debug GUI seen in figure 2.5. From here a lot of information can be obtained.

In the upper left corner all cameras and their process steps can be seen. These can be selected and then poped to the grid, figure 2.6, by clicking *Pop Window*. One a window configuration in the grid is set it can be saved by clicking *Save grid configuration* under the Grid tab in the menu bar.

In the upper right corner one can find profiling information for every step in the pipe line. In bottom of the window is the system log. This displays messages from inside the system.

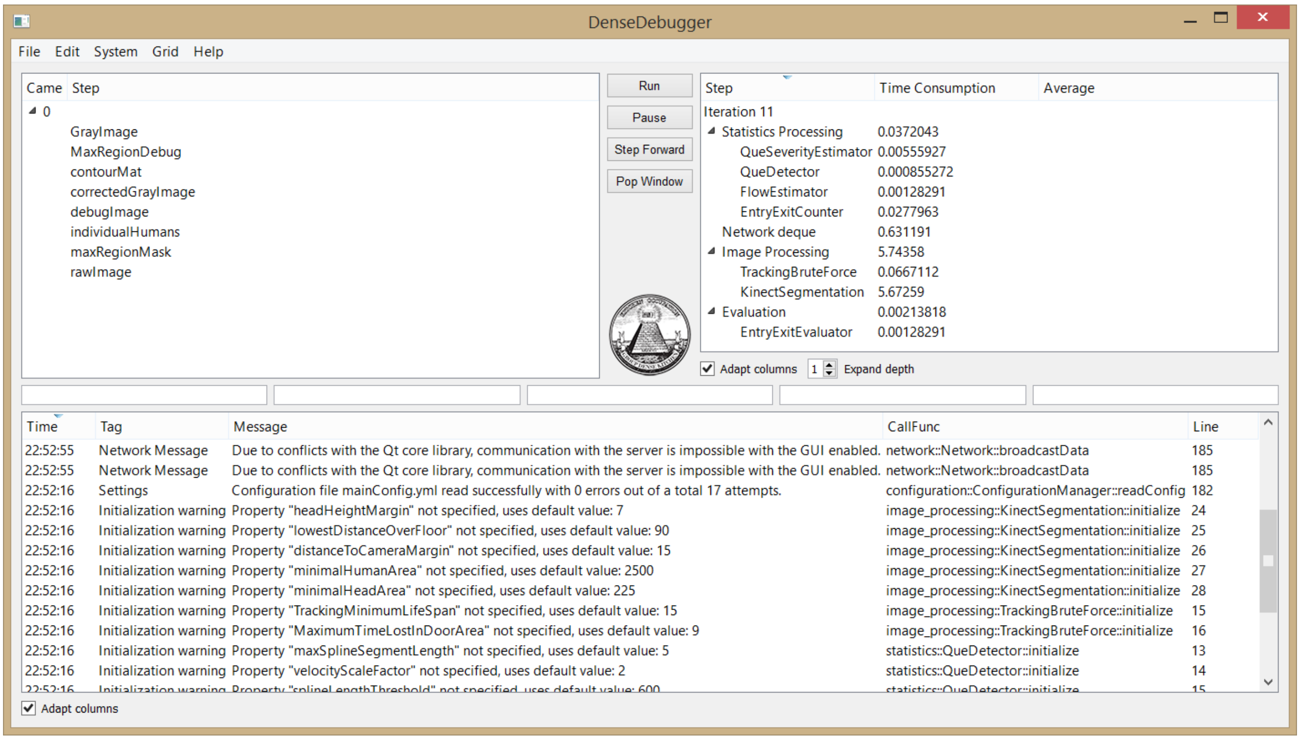

Figure 2.5: *The debug GUI.*

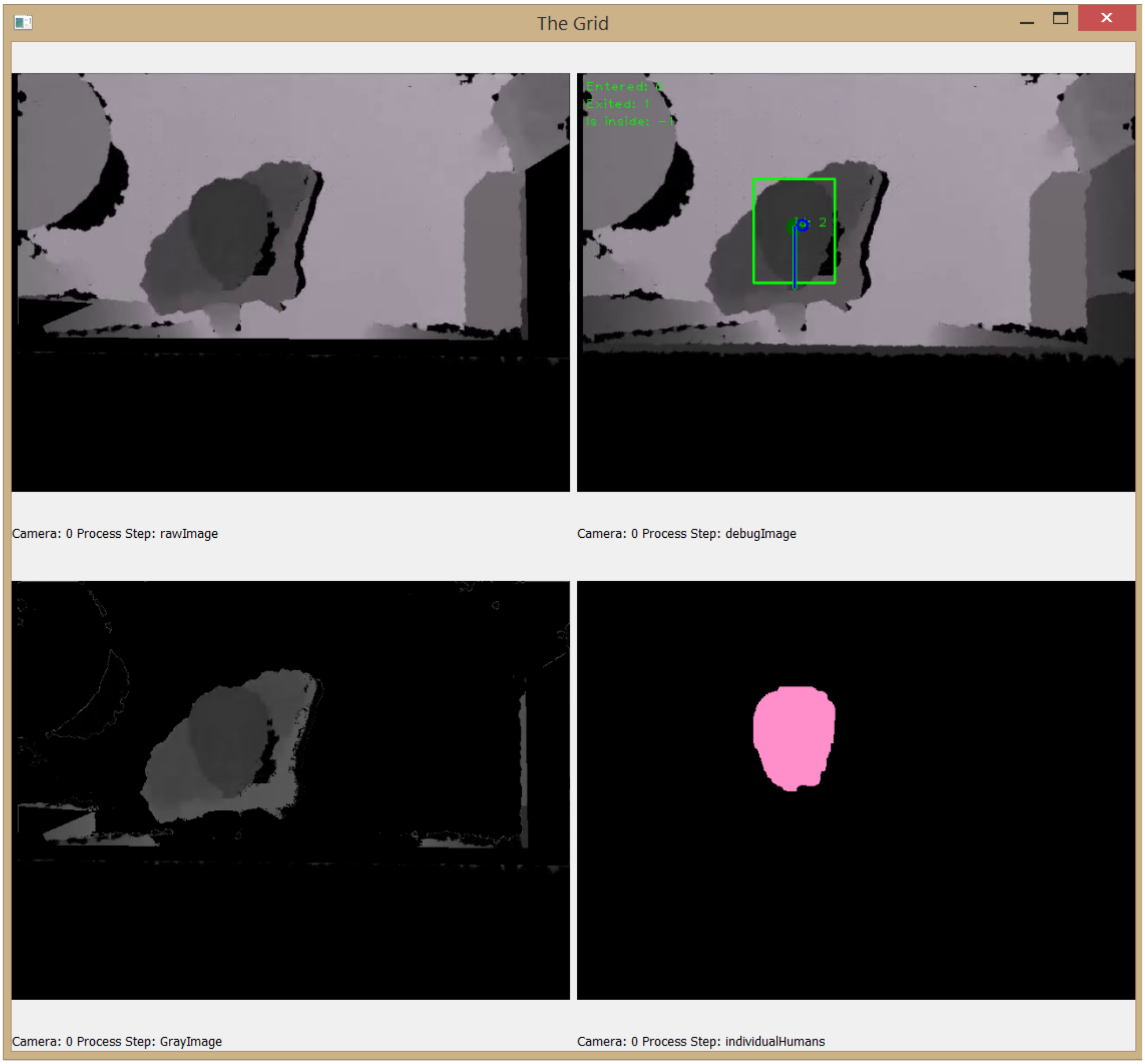

Figure 2.6: *The grid.*