CREST PROGRAM

Version 0.1

User's guide

Crest Software License Agreement

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

CHAPTER 1: INTRODUCTION

The Crest program is software to detect and track pedestrian in different scenarios and indoor and outdoor environments.

System Requirements

To use the Crest Program, is necessary an IBM-compatible PC. To effectively run the Crest Program it is recommended to have the following:

- 1 GB of RAM
- Dual core processor.
- Bumblebee2 stereo camera

CHAPTER 2: INSTALLING THE SOFTWARE

The Crest program is software available in two different versions to offer flexibility to the other user's programs.

Dynamic Library

This library can be added to an existent project. An example of how to interface with the DLL file it is available on DLLInterface.h file.

Crest Program

Minimal graphic interface. It provides the call to the camera and it executes the processing of the input data.

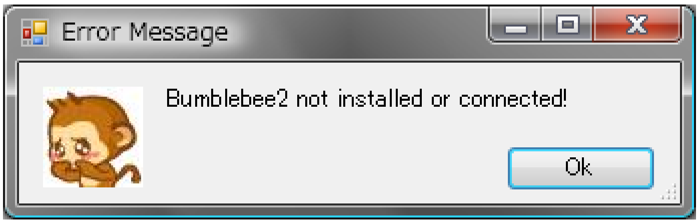
To install the Crest software

1. Run the CrestInstaller and select the destination folder.

CHAPTER 3: GETTING STARTED

How to use the GUI

Execute the program from the installation folder. If it will be not possible to run the program, an error message will appear (fig. 1).



Bumblebee2 not installed or connected!

Fig. 1 Example of error messages due to camera or configuration files problems.

If the program is properly installed and if there are no errors, a window form will appear (fig. 2).

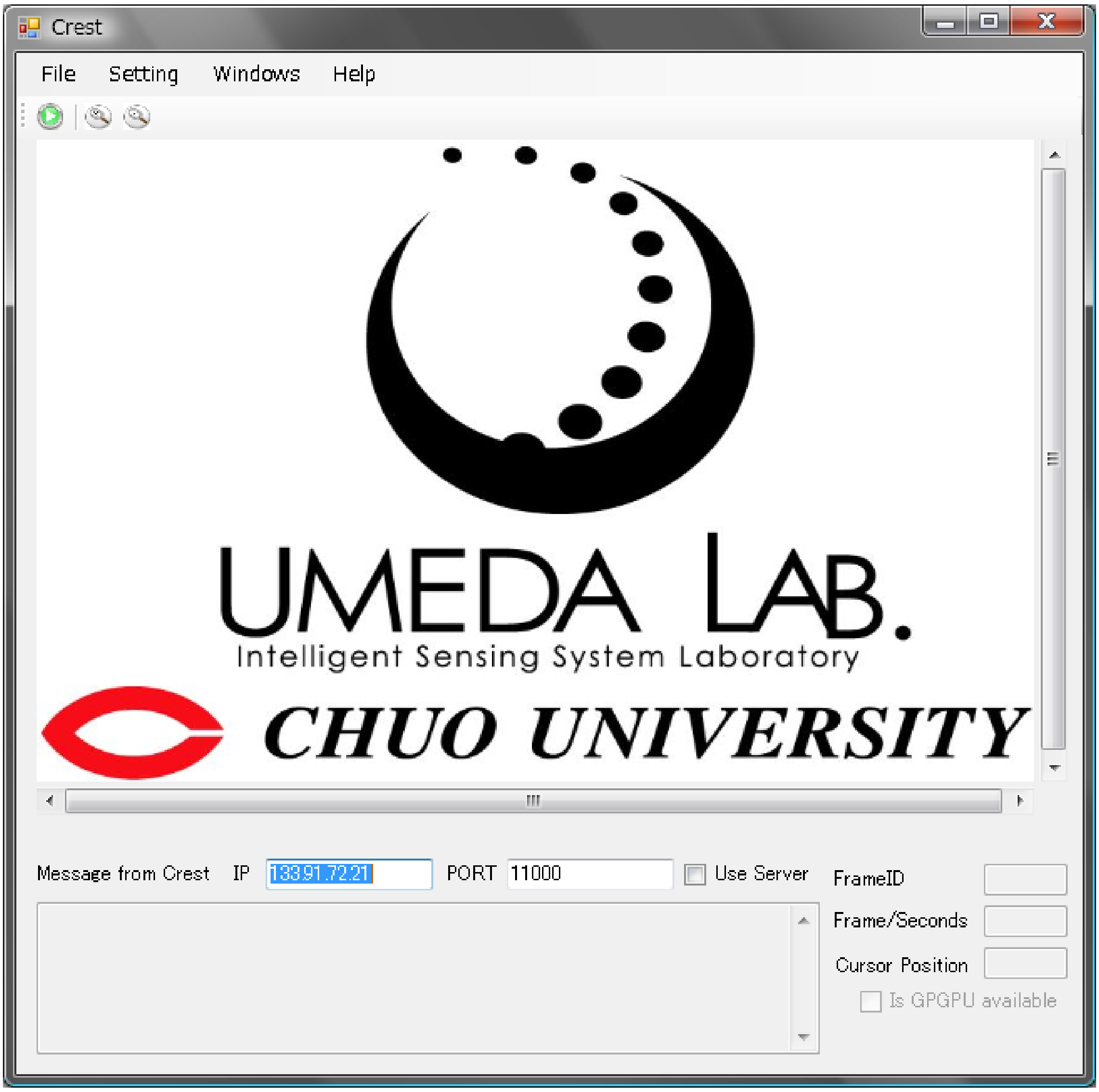
The menu is positioned on the top. Once Crest program is opened, only the setting menu will be available. The setting windows are used to modify the program parameters. The parameters can be adjusted only before the execution of the program. For a good performance, change the setting of the camera parameters (height and pitch).

On the bottom of the window, are given important information as FPS, and messages from the program as tracking information.

The checkbox “Use Server” allows the communication with a TCP/IP client. A demo version of the client is available. The program won’t work asynchronously and it will wait to communicate with the client.

In order to obtain the tracking data, it is necessary to check the “Use server” check box. A C# client is available on ConsoleApplication.zip.

The program uses the NVIDIA processor and GPU to update the background. If the GPU is available, a check will appear to the “Is GPGPU available” checkbox. If not available the background will be updated using CPU.



IP 133.91.72.21

PORT 11000

☐ Use Server

☐ Is GPGPU available

Fig. 2 Crest program main window form

Under the menu, the play button runs the program. The main display will show the rectified color acquired image. The pixel that will be analyzed will be highlighted in blue. Instead pixels considered shadow will be green (fig. 3).

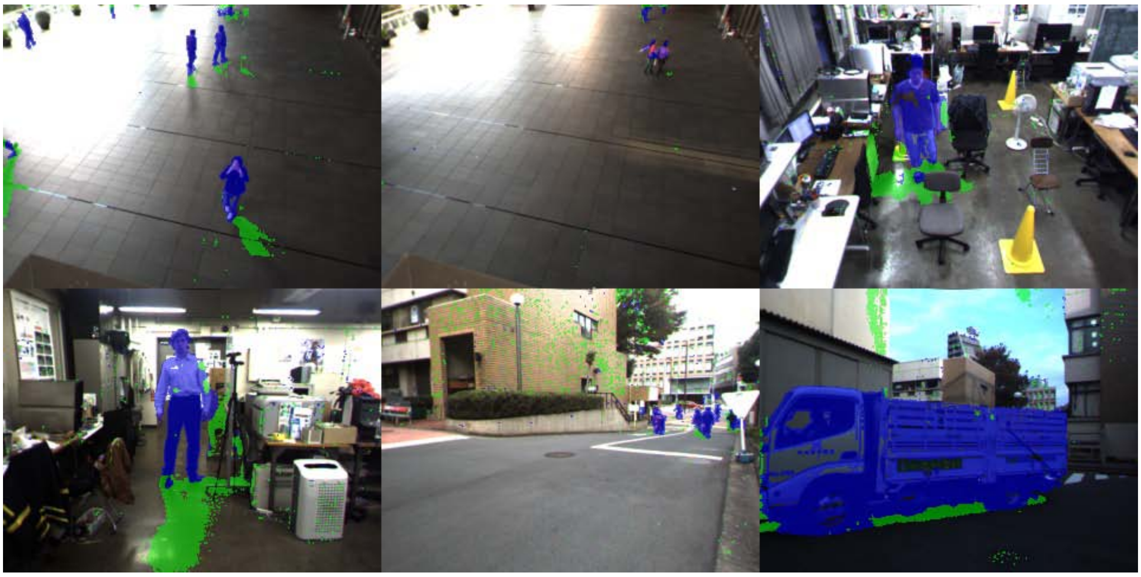
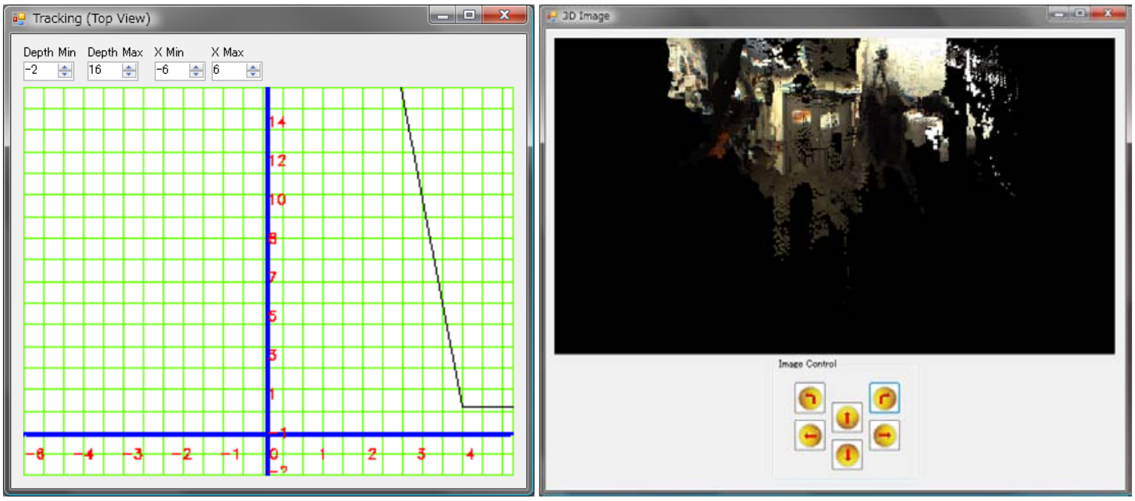


Fig. 3 Example of shadow and analyzed pixels highlighted on the image. In blue the used pixels, in green the detected shadow.

In run mode it is possible to display also the tracking result and 3D window. The windows are available on Windows menu. Tracking results and 3D view are shown in fig.4.



Depth Min -2
Depth Max 16
X Min -6
X Max 6

Fig. 4 Example of tracking and 3D view display.

CHAPTER 4: CREST DLL FUNCTION REFERENCE

This chapter presents a detailed description of each function in the DLL.

The functions are gathered in a C++ class style. In the follow a description of all the functions collected inside the DLL class.

IMPORTANT

All the pointers passed to the functions MUST be properly allocated before to make the call. The standard resolution of the images is 320x240 for rectified images and 640x480 for raw images.

CrestError

Some of the Crest functions return an error value that indicates whether an error occurred, and if it so what error. The following define type lists the kinds of errors that may be returned.

Declaration

```
#define CREST_NOERROR 0x00000000
#define CREST_ERROR_CAMERANOTINSTALLED 0x00000001
#define CREST_ERROR_NOCONFIGFILE 0x00000101
#define CREST_ERROR_NOPARAMETERSFILE 0x00000102
#define CREST_ERROR_CUDANOTINITIALIZED 0x00000201
```

Elements

CREST_ERROR_CAMERANOTINSTALLED	Unable to detect the camera
CREST_ERROR_NOCONFIGFILE	Unable to load the configurations file
CREST_ERROR_NOPARAMETERSFILE	Unable to load the parameters file
CREST_ERROR_CUDANOTINITIALIZED	Unable to initialize CUDA

LoadConfigFiles

Load the configuration and parameters files. Return an error if it occurs.

Declaration

```
virtual int LoadConfigFiles();
```

Initialize

Initialize the Crest internal structures and allocate the memory.

Declaration

virtual int Initialize();

InitializeCuda

Initialize the structures and allocate the memory in order to use GPGPU. Return an error code in the case of failure.

Declaration

virtual int InitializeCuda();

GrabImage

Grab a single image from the stereo camera. Return true in case of success.

Declaration

virtual bool GrabImage(int **tmp[2], unsigned char ***rgb, unsigned short **depth_map, unsigned short **depth_map_original, unsigned char **image_rectified);

Elements

tmp	Left and Right raw image (B/W)
rgb	Rectified color image
depth_map	Depth map
depth_map_original	A copy of the depth map
image_rectified	Rectified B/W image

Process

Provide to elaborate a single frame captured. Return true in the case of success.

Declaration

virtual bool Process();

GetCurrentOutputImage

Copy the current output image in a matrix. Return true in the case of success.

Declaration

virtual bool GetCurrentOutputImage(unsigned char ***rgb);

Elements

rgb	Rectified color output image

GetCurrentTrackingImage

Copy the current tracking image represented by a top view map. The scale of the image is proportionate to the dimension. Return true in case of success.

Declaration

virtual bool GetCurrentTrackingImage(unsigned char ***rgb, int depth_min, int depth_max, int xmin, int xmax);

Elements

rgb	Rectified color output image
depth_min	Min range of the depth in the display
depth_max	Max range of the depth in the display
ymin	Min x position in the display
xmax	Max x position in the display

GetNumMessages

Return the number of messages that the Crest program has in queue. The queue is cleared any time the function Process is called.

Declaration

virtual int GetNumMessages();

GetMessage

Copy the indicated message.

Declaration

virtual void GetMessage(char *msg, int num_message);

Elements

msg	Copy of the message
num_message	ID of the message the user wants to copy.

GetNumTrackingMessages

Return the number of tracking messages the Crest program has in queue. The queue is cleared any time the function Process is called

Declaration

virtual int GetNumTrackingMessages();

GetTrackingMessage

Copy the indicated message.

Declaration

virtual void GetTrackingMessage(char *msg, int num_message);

Elements

msg	Copy of the message
num_message	ID of the message the user wants to copy.

GetWorldCoordinate

Memorize in a vector the information relative to the image captured by the camera. The size of the vector must be proportionate to the resolution.
The vector will contain 6 values for each pixel:
X coordinate, Y coordinate in the world coordinate, Z coordinate in the world coordinate, red, green, and blue.

Declaration

virtual void GetWorldCoordinate(float *world_coordinate, float resolution);

Elements

world_coordinate	Vector contains information of the image like color and position in the world coordiante.
resolution	Image resolution [0, 1].

GetCurrentFrameNumber

Return the current internal frame number.

Declaration

```
virtual int GetCurrentFrameNumber();
```

GetTrackInformation

Copy the tracking information. An object that it is just detected but not tracked will have a negative ID.

Declaration

```
virtual void GetTrackInformation(int *x, int *y, float *X, float *Y, float *Z, int *ID, const int iMaxSize, int &iFoundObjects);
```

Elements

x	Current x position in the image
y	Current y position in the image.
X	Current X position in the world coordinates.
Y	Current Y position in the world coordinates.
Z	Current Z position in the world coordinates.
Id	ID of the tracked object.
iMaxSize	Max number of objects can be copied.
iFoundObjects	Total number of found and tracked objects.

CHAPTER 5: CONCLUSION

Current Limitations for the program and DLL

Can be used only under windows OS, 32bit. The working systems known are WindowXP and Vista.
Only Bumblebee2 camera can be used and requires Bumblebee2 drivers installed.

In the near future

Increase the support to other stereo vision system.
Increase performances. Offer multi-threading support.
Offer a better 3D navigation system.
Increase the compatibility with other OS.

CONTACT INFORMATION

For any questions, concerns or comments please contact us via the following method:

Email:
crest@sensor.mech.chuo-u.ac.jp