A Manual for

altURI – An Alternative

USARSim Robot Interface

UNIVERSITY OF
LINCOLN

Centre for Vision and Robotics Research

Author	Mark N R Smith
Approved	TBA
Current Version	
Issue Date	
Review Date	

Document Control

Revision History

Version	Date	Author(s)	Notes on Revisions
0.1	DD.MM.YY	Author	Initial draft

Contents

1 Introduction

This manual describes a software system designed to be used in vision and robotics research at the University of Lincoln and elsewhere. The software provides modules to receive images and sensor information from and send commands to virtual robots within a USARSim environment.

The software provides simple and configurable interfaces to any number of simulated robots using a combination of web and Application Programmer Interface mechanisms.

This allows simulated robots to be controlled by a wide range of application software such as MATLAB, C++ and Java. All components of the altURI system are licensed as Open Source.

The USARSim (Unified System for Automation and Robot Simulation) system was created to provide a realistic low cost simulation tool for robots in real environments. It is based on the commercial games platform Unreal Tournament (Epic Games, 2004, 2009) which is customised to provide models of various robots and environments. Additional software components are provided to support image and sensor data acquisition and implement robot actions.

2 altURI Requirements

The altURI software was designed to control robots in the USARSim robot simulator running on the Unreal Tournament game. It consists of three components plus a test harness. Two of the components are Windows Dynamic Link Libraries (DLLs); one sends commands and the other acquires images from the USARSim game environment. A small utility program is provided to load the image acquisition DLL and start the USARSim game program. A test harness is provided that can be used to view images and sensor data as well as send control commands using buttons on the test program.

2.1 Background

During research work the USARSim robot simulator was proposed for developing a robot control application during 2009. However the USARSim project was perceived to suffer from some disadvantages:

1. The system was moving to Unreal Tournament 3 (from UT2004) and all the required new modules were not yet in place;

2. A control system interface to general software application was not present and the existing control applications appeared to be fairly specialised;

3. The image acquisition system used in UT2004 did not appear to work well;

4. The overall future of the USARSim system was not clear.

2.2 Design goals

The altURI system was conceived to address these issues and provide additional functionality. The design goals were:

1. To support image acquisition from UT2004, UT3 and any other virtual environment using DirectX8, DirectX9, DirectX10, DirectX11 or OpenGL graphics.

2. To provide a generalised configurable control system allowing simple interface to any application.

3. To allow image acquisition of configurable dimensions over the web.

4. To be fully Open Source.

5. To support UT3 and other games using the Steam (or any) content delivery system.

6. To become a part of the USARSim project.

The issue of image acquisition from UT3 has subsequently been addressed by the release UPIS software.

3 altURI Architecture

3.1 altURI Components

The four components of the system altURI are:

altURI_Hook.dll – a 32-bit Windows DLL that loads into a graphics environment such as UT2004, UT3. The hook DLL provides a cut down HTTP web server to distribute images. It also provides an Application Programmer Interface (API) similar to the original USARSim hook module and similar to the UPIS system.

altURI_LoadDLL.exe – a 32-bit Windows command line program that loads altURI_Hook.dll into the environment of a graphics program.

altURI_Cmd.dll – a 32-bit Windows DLL that provides an Application Programmer Interface (API) to functions that sends commands to the USARSim game environment and receives remote images via HTTP.

altURI_UI.exe – a 32-bit Windows test program that uses altURI_Hook.dll and altURI_Cmd.dll to receive images and sensor data and send commands. Allow testing of robot commands and sensor information configuration.

3.2 altURI Component Context

Figure 1 below shows the context of the altURI components .

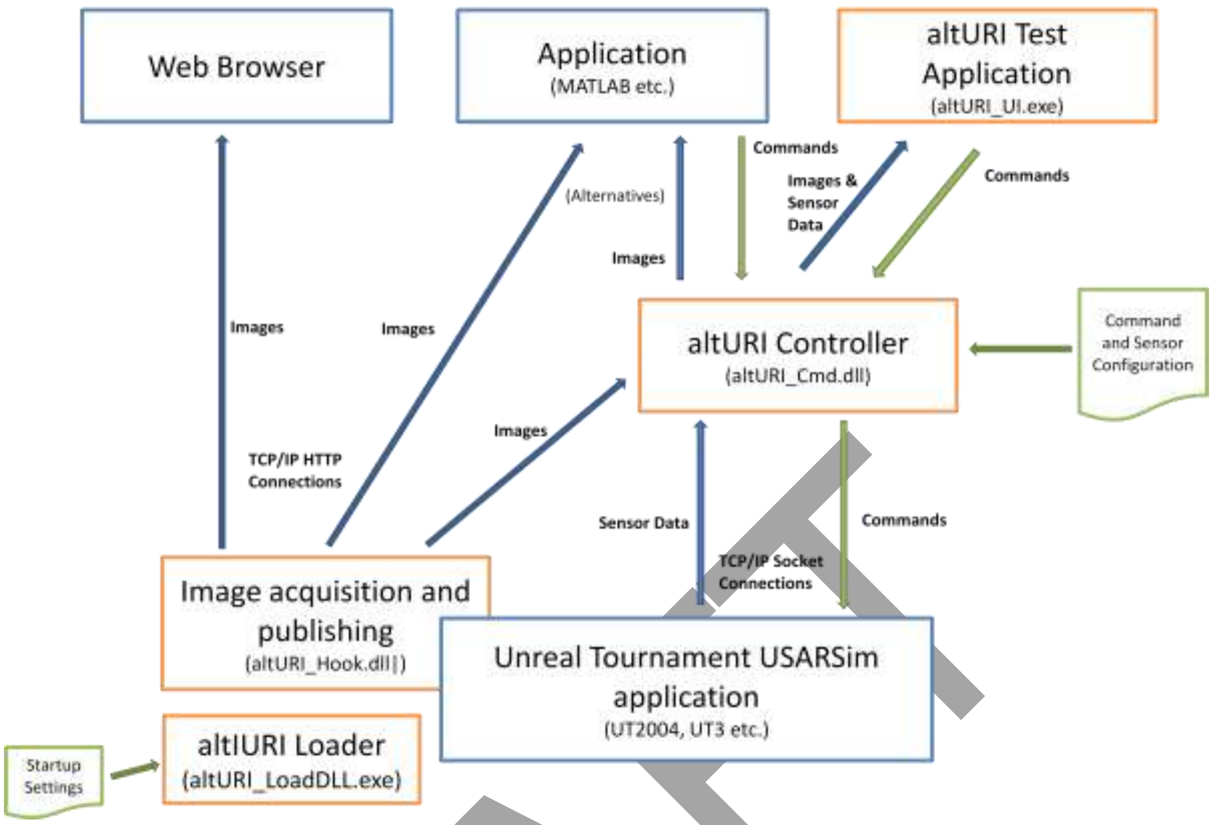

Figure 1 – altURI Components

4 Implementation

4.1 Development

The software components of altURI are written in for the Microsoft Windows platform using Microsoft Visual C++ 9.0. The graphics routines use the OpenCV 2.0 Library for image handling. The DLL runs a simple web also supplies an API that returns OpenCV IPL images and supports backwardly compatible formats.

The altURI LoadDLL program is a command line console application that starts the game application and loads the hook DLL. Any DLL can be loaded into any application that allows it.

uses Microsoft Foundation Classes (MFC) (ref) for window handling. Therefore the distribution requires the OpenCV and MFC run time components which are packaged in a setup program.

4.2 Third Party Components

The altURI image acquisition method is based on the Open Source Taksi application to intercept DirectX calls rather than the proprietary Microsoft Detours for hooking. This allows porting to 64-bit architecture and avoids Detours licensing issues allowing altURI to be completely Open Source.

Image handling throughout altURI uses the OpenCV 2.0 library

The altURI Test application uses a dialog (window) handling module based on a Code Project library (http://www.codeproject.com/KB/dialog/RPResizeDlg.aspx) which has no license restrictions. The test application and It also uses configuration/ini file library (http://www.codeproject.com/KB/cpp/cinifileByCabadam.aspx?msg=15460) from Code Project with no license restrictions.

4.3 Web Vision System

The altURI Hook component contains a small dedicated web server that will supply images and a web page to browsers or other HTTP compatible software. The images can be supplied as JPEG or PNG images which can be seen in most web browsers. An HTML page with a Javascript refresh script can also show continuous motion

The image provided is a snapshot of the current image in the USARSim (or other) graphics environment which allows applications to obtain compressed images (lossless with png) remotely over a network or the internet.

With no parameters the image is returned as the same size as the USARSim image is at that moment. The following URL parameters can be used to control the image returned:

4.4 Web Parameters

Assuming altURI is running on a machine with a DNS name of MyaltURI.com the URL could be:

http://www.MyaltURI.com/image.jpg?h=300&w=400&x=100&y=200&q=50

where:

h is the image height in pixels
w is the width in pixels
x is the x offset in pixels – allows a partial (region) of the image to be returned
y is the y offset in pixels – allows a partial (region) of the image to be returned
q is the percentage compression attempted (or level for PNG)

The URL is not case sensitive and any image name can be used as its extension (ie. .jpg, png etc.) is supported. If a parameter is not supplied or outside a reasonable range the software uses a sensible default.

A web page can also be returned containing an image using the same parameters with one addition:

http://www.MyaltURI.com/image.html?h=300&w=400&x=100&y=200&q=50&t=3

This has the same effect on the image as above but uses some javascript to refresh the page every t seconds. This allows a simple 'movie' display using a jpeg image (but is not streaming).

4.5 Robot Control

The altURI Command system uses a Windows DLL API interface as a standard ‘C’ DLL. Functions are exported in the DLL to make them available to most applications that load the library.

The command parameters are configured using an text (.ini) file to set minimum and maximum and increment numeric values that are placed in the command string when sent to USARSim. A single command with a value can be sent or all commands can be sent simultaneously. The configuration file also specifies the IP address and port number to connect to the USARSim environment.

The command system is intended to provide maximum flexibility whilst maintaining a simple API interface. In USARSim when a robot is given a command it continues until it receives another which overwrites the first.

For example, if commanded to drive forward then commanded to drive left it will drive left after the second command. So a command to drive diagonally forward to the left must specify forward and left simultaneously. Each command in altURI is assigned a drive number and a maximum of twelve commands can be defined (although this could easily be increased). The drive commands are expressed in increments from the current value which is zero when nothing is moving.

Control functions

The commands either return a true/false (1/0) for success/fail or the numeric value requested.

The configuration file is loaded for a specific robot/configuration using:

```
SetupControl( FileName );
```

Any raw USARSim command text string can be sent with the SendRobot command which is commonly used to create (spawn) a robot:

```
SendRobot( szAnyCommand );
```

The drive command is

```
CommandDrive( DriveNumber, NumberofIncrements )
```

If NumberofIncrements is zero no change is made (so there is no point in sending this command)

The current value and minimum, maximum and increment values respectively can be obtained by using:

```
GetDriveCurValue( DriveNumber )
GetDriveMinValue( DriveNumber )
GetDriveMaxValue( DriveNumber )
GetDriveIncValue( DriveNumber )
```

All drive commands with increments for up to twelve drive commands can be sent as:

```
CommandDriveAll(  iIncrements1,
                  iIncrements2,
                  iIncrements3,
                  iIncrements4,
                  iIncrements5,
                  iIncrements6,
                  iIncrements7,
                  iIncrements8,
                  iIncrements9,
                  iIncrements10,
                  iIncrements11,
                  iIncrements12 );
```

Similarly of the increment value is zero the current value is used – not change – which allows a complex command to be sent.
The command below stops all motion and sets all current values for defined drive commands to zero:

```
CommandDriveAllStop()
```

Similarly to Drive commands any number of other commands can be defined in the configuration file to be used singly using the function:

```
CommandOther( CommandNo, iIncrements )
```

Any number of sensor messages can be defined in a similar format to other commands. The sensor commands define what will be retrieved from the sensor messages sent out by a robot in the USARSim environment.

The following command determines if sensor messages are caught by the altURI program:

```
SetCatchMessages( bCatchMessages )
```

If the value is 1 – true the messages are caught or zero they are not.

The following commands enable an application to set up an array to receive a number of sensor values:

```
GetNumberMessages( )
```

This command determines the number of values currently waiting to be retrieved. The calling application can then allocate data storage large enough to receive them by the following command:

```
GetMessagesArray( dArray2d[] )
```

The array defined returned the number of values as columns and each value received as rows. [to be clarified].

The application can also receive either all the raw messages, or just those with the sensor tags defined in the configuration file (see below) - or neither, that are returned from the USARSim environment using the command:

```
SetCatchReceivedLines( bCatchLines, bCatchMessageLines )
```

Again if the value is 1 – true those messages are caught or zero they are not. The command below retrieves a single message from the queue held once the capture is switched on with `SetCatchReceivedLines`:

```
string GetReceivedLine( )
```

Finally the session must be terminated using the command:

```
CloseControl( )
```

Image acquisition [TBA…]

The 'c' include file that specifies these functions is made available to facilitate interface programming.

Control Configuration Setup (.ini file)

An example configuration file is shown below. Each of the three sections has a header that specifies how many sub entries are defined. For example the file below specifies four Drive commands and supplies four Drive sections. When commands require numeric parameters the format is specified as a 'c' printf format (ref). So %1.2f specifies a floating point number with one digit to the left of the decimal and two to the right.

```
; altURI Command Settings
;
; lines prefixed with hash or semicolon are ignored as comments
;
[connection]
SimIP4=127.0.0.1
SimPort=3000

; global drive settings
[drives]
DrivePrefix=DRIVE
DriveTotal=4
; changes Min to -100 and Max to 100
DriveNormalize=0

; Height controls
[drive1]
Key={AltitudeVelocity %1.2f}
Min=-5.00
Max=5.00
Increment=0.01

; Forwards backwards controls
[drive2]
Key={LinearVelocity %1.2f}
Min=-5.00
Max=5.00
Increment=0.01

; Left right controls
[drive3]
```

```
; note: numerical value with printf formatting - in this case one digit and two
decimals
Key={LateralVelocity %1.2f}
Min=-5.00
Max=5.00
Increment=0.01

; Rotation controls
[drive4]
Key={RotationalVelocity %1.2f}
Min=-5.00
Max=5.00
Increment=0.01

; global other non drive command settings
[commands]
CommandPrefix=
CommandTotal=1

; other command one
[command1]
Key=BLAH {Something %u}
Min=0
Max=1
Increment=1

[messages]
MessageTotal=1

; one entry per float value (even in the same message line)
[message1]
MessageName=SEN
KeyName=Location
Index=0
```

Testing

altURI Hook, the image acquisition has been tested and found to work on Windows XP, 2003, Vista and Windows 7. The UT2004 and UT3 games have been shown to work - as well as other potential simulator environments such as Quake (OpenGL engine), Crysis (CryEngine) and Half-Life, CounterStrike, L4D (Source Engine).

Similarly altURI_Cmd and altURI_UI, the command module and test application, have been tested on the same versions of Windows as well as on UT2004 and UT3. It has not been tested in other simulator environments.

The web component of the altURI image acquisition has been tested on IE

MATLAB Interface

The altURI system command module has been designed to allow application with a programming interface such as MATLAB to use it.

An example of a MATLAB script