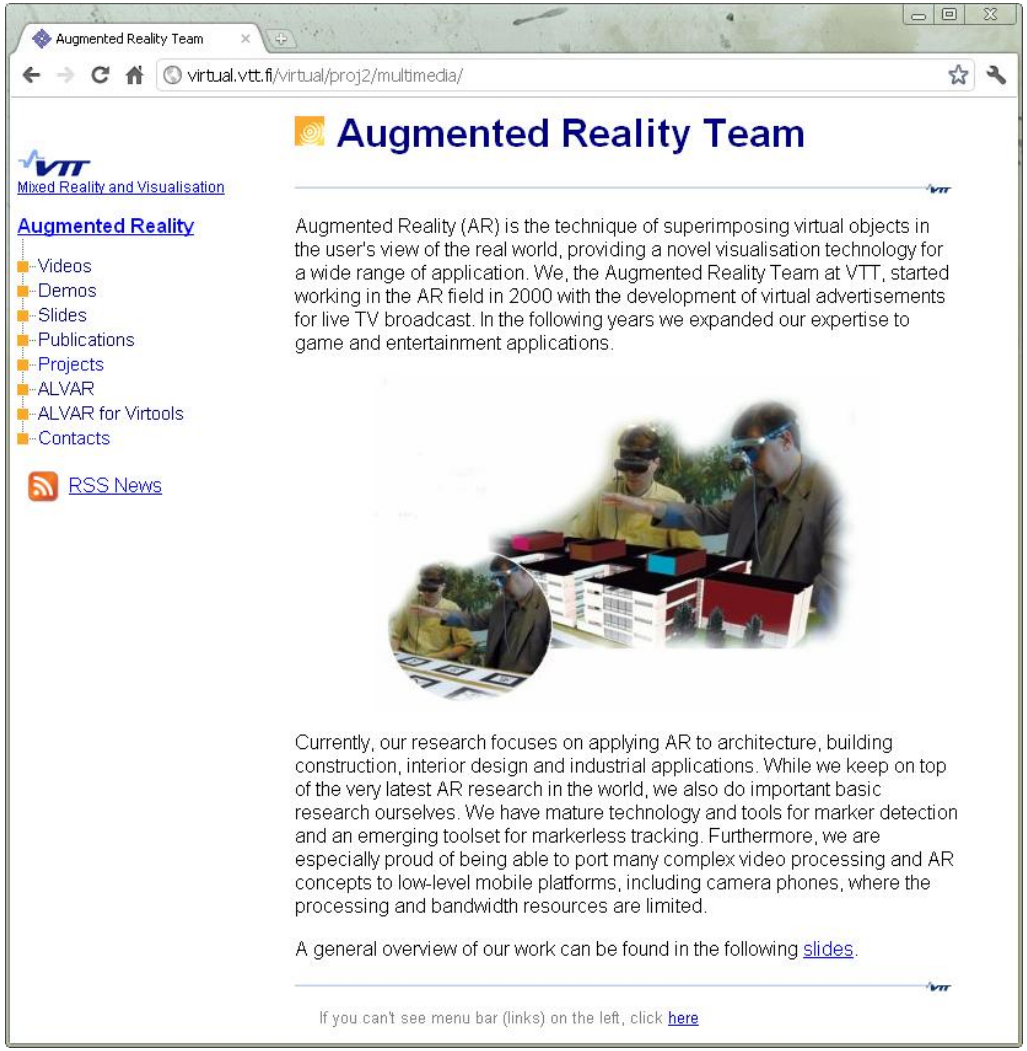
ALVAR

A Library for Virtual and Augmented Reality

VTT Augmented Reality Team

- The VTT AR Team web page: http://www.vtt.fi/multimedia
- Contains a slide presentation explaining Augmented and Mixed Reality, downloadable demo apps, and demo videos.
- A large part of the team's AR know-how (but not all of it) is embodied in the ALVAR open-source AR library: A Library for Virtual and Augmented Reality



ALVAR (Desktop)

A Library for Virtual and Augmented Reality

- Subroutine library originally developed by VTT
- Further development in co-operation with e.g. Aalto Univ., Nokia, Columbia Univ.

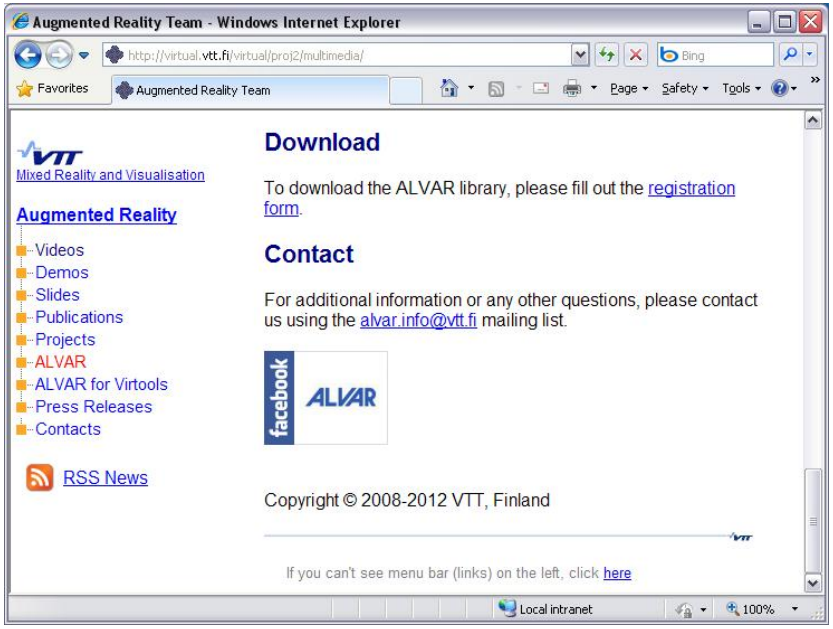
Versions

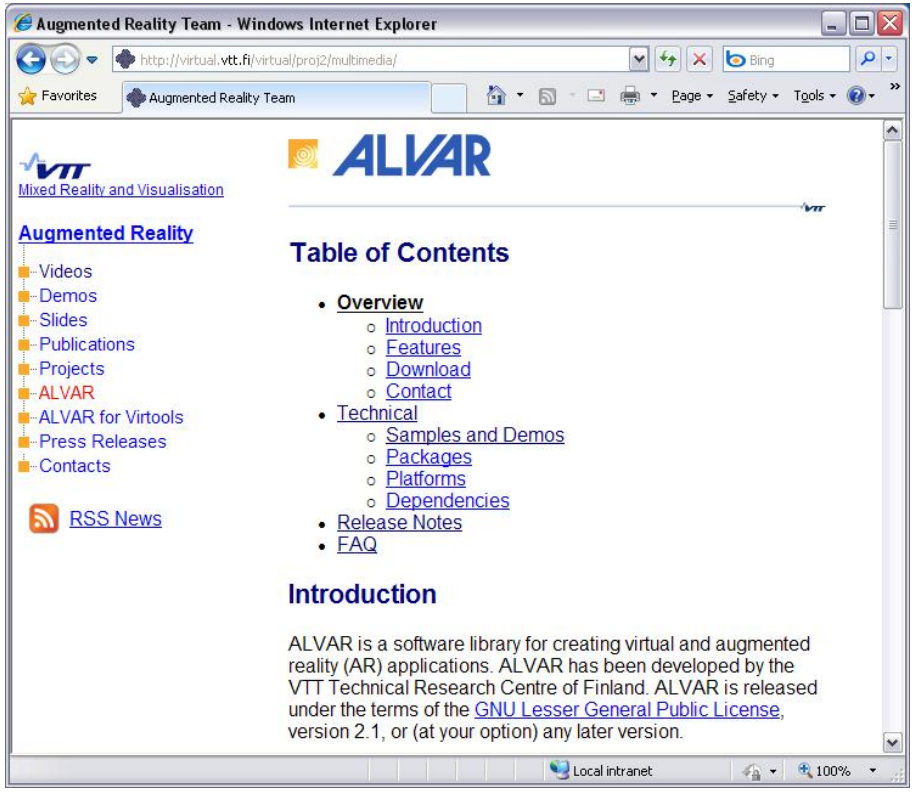
- First versions of ALVAR were released in 2009. Earlier ALVAR was divided into Basic (core features) and Pro (advanced features) versions
- But the Basic and Pro versions have been merged into the ALVAR Desktop version 2.0.
- ALVAR Desktop version 2.0 was released in May 2012 as open-source software; this presentation describes v.2.0

- http://www.vtt.fi/multimedia/alvar.html

ALVAR: Web Page Contents

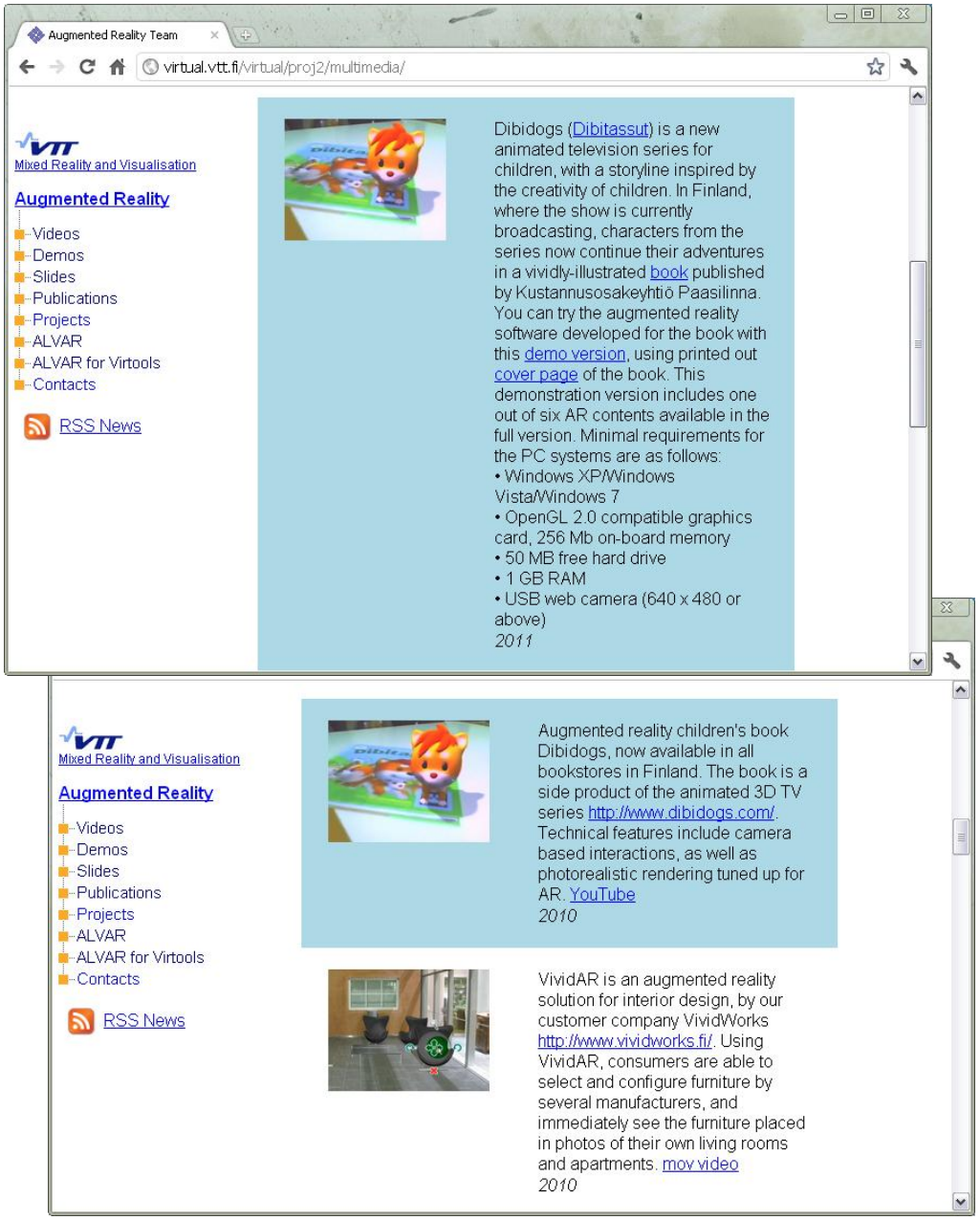
- The VTT AR Team web page: http://www.vtt.fi/multimedia
- Instructions of how to obtain ALVAR
- Separate binary distributions available, including using ALVAR with OSG





ALVAR Desktop Demo

- The VTT AR Team web page: http://www.vtt.fi/multimedia
- Most Demos are programmed using ALVAR Mobile
- But the Dibidogs demo use only ALVAR Desktop (and 3D models with animations)
- There is also a demo video of the Dibidogs program
- Also the VividAR demo video presents a program that has been programmed using ALVAR Desktop



ALVAR Introduction (1/2)

ALVAR is a software library for creating virtual and augmented reality applications. ALVAR has been developed by the VTT Technical Research Centre of Finland.

The current version of the library mainly supports marker-based augmented reality applications, but also includes tools for markerless augmented reality.

ALVAR is designed to be as flexible as possible. It offers high-level tools and methods for creating augmented reality applications with just a few lines of code. The library also includes interfaces for all of the low-level tools and methods, which makes it possible for the user to develop their own solutions using alternative approaches or completely new algorithms.

ALVAR Introduction (2/2)

ALVAR is currently provided on Windows and Linux operating systems and requires only one third party library (OpenCV). ALVAR is independent of any graphical libraries and can be easily integrated.

On the other hand, this implies that ALVAR itself contains no support for 3D graphics or 3D models – these must be implemented using other software libraries.

There are separate demo programs, which illustrate how to use ALVAR with OpenSceneGraph for 3D graphics. The binaries of these demo programs are included in the freely downloadable ALVAR **Bin** distribution (see slide 13).

ALVAR Desktop License

ALVAR Desktop license is the GNU LGPL v.2.1, so it is free for both commercial and non-commercial use.

ALVAR 2.0 is distributed under the terms of the GNU Lesser General Public License (LGPL) version 2.1 or later. The license terms can be found at http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html. By downloading ALVAR 2.0, you agree to be bound by the terms of the GNU LGPL version 2.1 or later.

Other versions of ALVAR (e.g. ALVAR Mobile, ALVAR Web) are commercial.

ALVAR Features (1/4)

- Detecting and tracking 2D markers. Currently two types of square matrix markers are supported. Future marker types can easily be added. ALVAR keeps the marker pose estimation as accurate as possible. Furthermore, ALVAR uses some tracking heuristics to identify markers that are "too far" and to recover from occlusions in the multimarker case for example.
- Using a setup of multiple markers for pose detection. The marker setup coordinates can be set manually or they can be automatically deduced using various methods.

ALVAR Features (2/4)

- Tools for calibrating a camera. Distorting and undistorting points, projecting points and finding exterior orientation using point-sets.
- Hiding markers from the view.
- Several basic filters: average, median, running average, double exponential smoothing. Kalman filters for sensor fusion: Kalman filter, extended Kalman filter and unscented Kalman filter.
- Several methods for tracking using optical flow.
- Markerless tracking using the SfM and Fern's algorithms.

ALVAR Features (3/4): Marker-Based Tracking

- Detecting and tracking 2D markers
  - Two types of markers
  - Future marker types can be added
  - Accurate pose estimation
  - Heuristics for intelligent tracking
- Multiple marker setups are supported
  - Marker relations set manually or detected automatically
  - Recover from occlusions
- Hiding markers from the view
- Computer vision tools
  - Calibrating cameras
  - Distorting and undistorting points
  - Projecting points

ALVAR Features (4/4): Markerless Tracking

- Markerless tracking via SfM using image features
  - Uses marker for initialization
- Image-based markerless tracking
  - Assumes a planar target
  - Can use any feature detector
  - Based on Fern's classifier for matching
- Several methods for tracking using 2D optical flow
  - PSA, extended PSA for rotation
  - Image feature tracking
  - Statistical tracking
- Several basic filters
  - Average, median, running average, double exponential smoothing
  - Kalman filter, EKF, UKF

The 3 ALVAR Packages

There are three different ALVAR distribution packages, which all can be freely downloaded:

- **Bin** – for those who only want to test AR and ALVAR; contains precompiled binary versions of the ALVAR samples and OSG demos; no C++ compiler nor 3rd-party libraries are required
- **Sdk** – for those who want to build their own AR applications using ALVAR; contains ALVAR header files, precompiled ALVAR libraries, and HTML documentation; a C++ compiler and 3rd-party libraries are required
- **Src** – for those who want to compile ALVAR themselves; contains the raw source code of ALVAR; a C++ compiler and 3rd-party libraries are required

Please note that the 3 distributions are NOT subsets of each other, e.g. HTML documentation is in the **Sdk** distribution only. This ALVAR Presentation and the User's manual must be downloaded separately

ALVAR Requirements (Sdk package)

ALVAR has been tested with the following environments:

- Windows XP 32-bit; Microsoft Visual Studio 2005, 2008 and 2010 (versions 8, 9 and 10).
- Linux 32-bit and 64-bit; gcc versions 4.3, 4.4, and 4.5

ALVAR core library requires the following 3rd party library:

- OpenCV 2.4.0

ALVAR sample code requires:

- GLUT 3.7.6
- CMake 2.8.3

The separate demo programs require:

- OpenSceneGraph 2.8.4

(The **Src** distribution requires both OpenCV 2.4.0 and CMake 2.8.3)

ALVAR Sample Code (1/3)

- ***SampleCamCalib*** – This is an example of how to use *ProjPoints* and *Camera* classes to perform camera calibration using a chessboard pattern.
- ***SampleCvTestbed*** – This is an example of how to use the *CvTestbed*, *CaptureFactory* and *Capture* classes in order to make quick OpenCV prototype applications.
- ***SampleFilter*** – This is an example of how to use various filters: *FilterAverage*, *FilterMedian*, *FilterRunningAverage*, *FilterDouble-ExponentialSmoothing*, *Kalman, KalmanEkf* and *FilterArray*.
- ***SampleIntegralImage*** – This is an example of how to use the *IntegralImage* and *IntegralGradient* classes for image gradient analysis.
- ***SampleLabeling*** – This is an example of how to label images using *LabelingCvSeq*.

ALVAR Sample Code (2/3)

- ***SampleMarkerCreator*** – This is an example that demonstrates the generation of *MarkerData* (or *MarkerArtoolkit*) markers and saving the image using *SaveMarkerImage*.
- ***SampleMarkerDetector*** – This is an example that shows how to detect *MarkerData* markers and visualize them using *GlutViewer*.
- ***SampleMarkerHide*** – This is an example that shows how to detect *MarkerData* markers, visualize them using *GlutViewer* and hide them with *BuildHideTexture* and *DrawTexture*.
- ***SampleMarkerlessCreator*** – This is an example of how to use the *FernImageDetector* class to train a Fern classifier for markerless image-based tracking.
- ***SampleMarkerlessDetector*** – This is an example of how to use the *FernImageDetector* and *FernPoseEstimator* classes to detect and track an image and visualize it using *GlutViewer*.

ALVAR Sample Code (3/3)

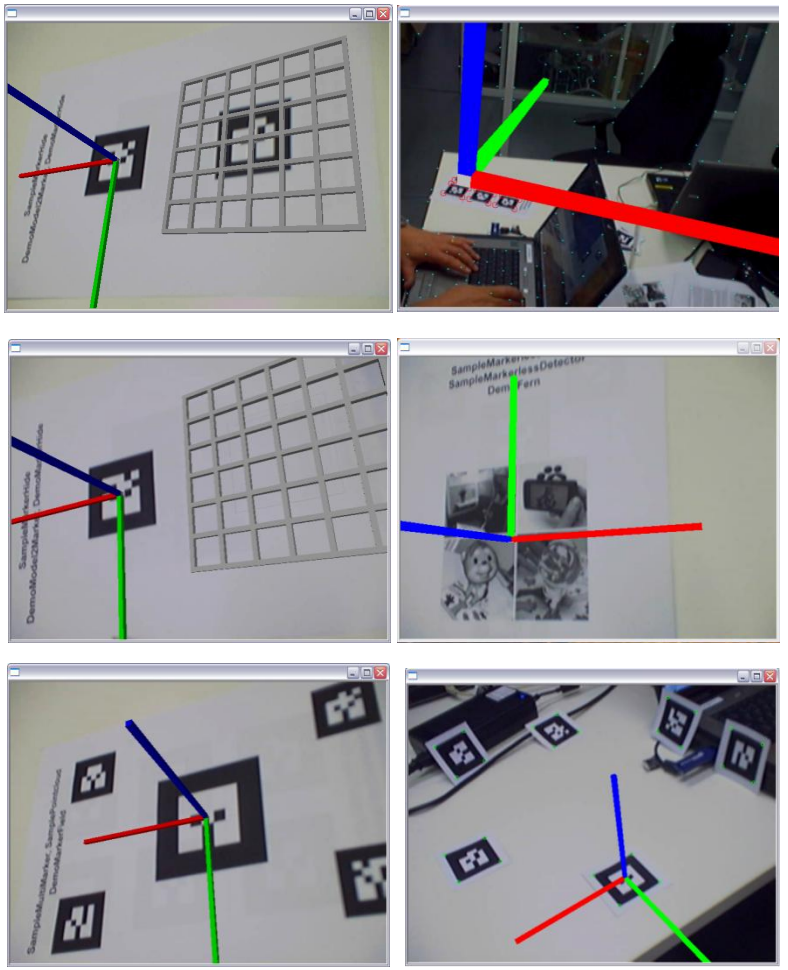
- ***SampleMultiMarker*** – This is an example that demonstrates the use of a preconfigured *MultiMarker* setup.
- ***SampleMultiMarkerBundle*** – This is an example that automatically recognises and optimizes *MultiMarker* setups using *MultiMarkerBundle*.
- ***SampleOptimization*** – This is an example of how to use the *Optimization* class by fitting curves of increasing degree to random data.
- ***SamplePointcloud*** – This is an example showing how to use *SimpleSfM* for tracking the environment using features in addition to *MultiMarker*.
- ***SampleTrack*** – This is an example that shows how to perform tracking of the optical flow using *TrackerPsa*, *TrackerPsaRot*, *TrackerFeatures*, *TrackerStat* or *TrackerStatRot*.

Dibidogs Application, using code mostly in SampleMarkerlessCreator and SampleMarkerlessDetector

Demo programs:
Using ALVAR with OpenSceneGraph

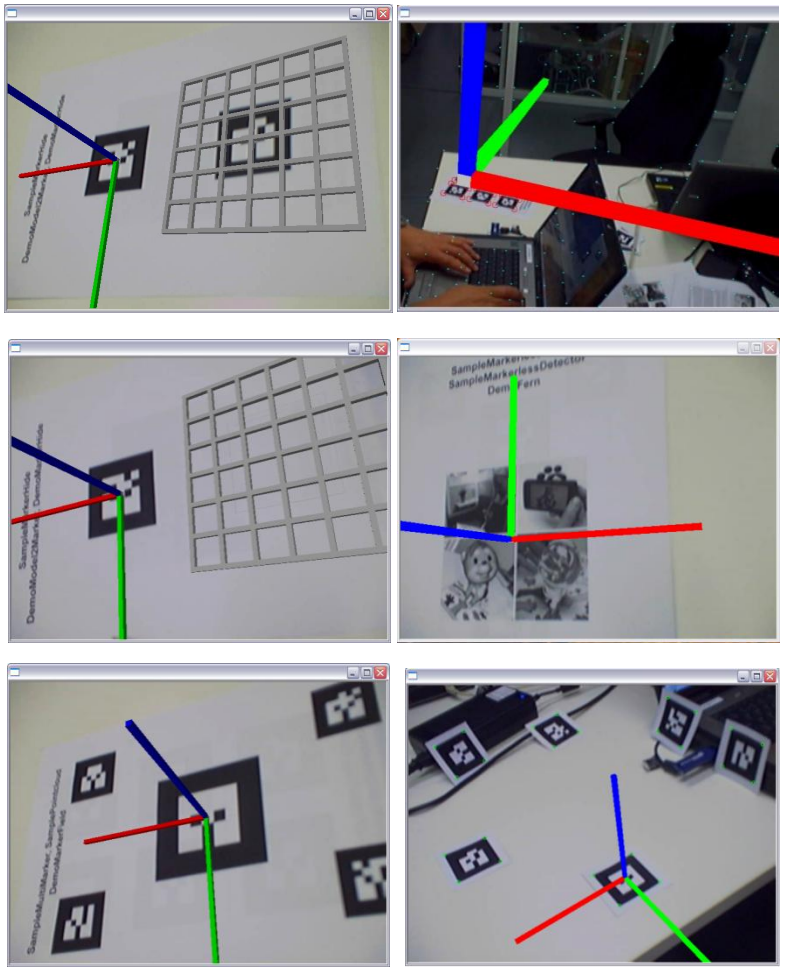
- ALVAR **Sdk** distribution package contains C++ source code only, the users must compile their own programs.
- ALVAR itself contains no support for 3D graphics or 3D models.
- There are separate demo programs, which use OpenSceneGraph for 3D graphics.

- *The **Bin** distribution package contains compiled binaries for Windows and Linux of the samples and demo programs. This package must be downloaded separately.*



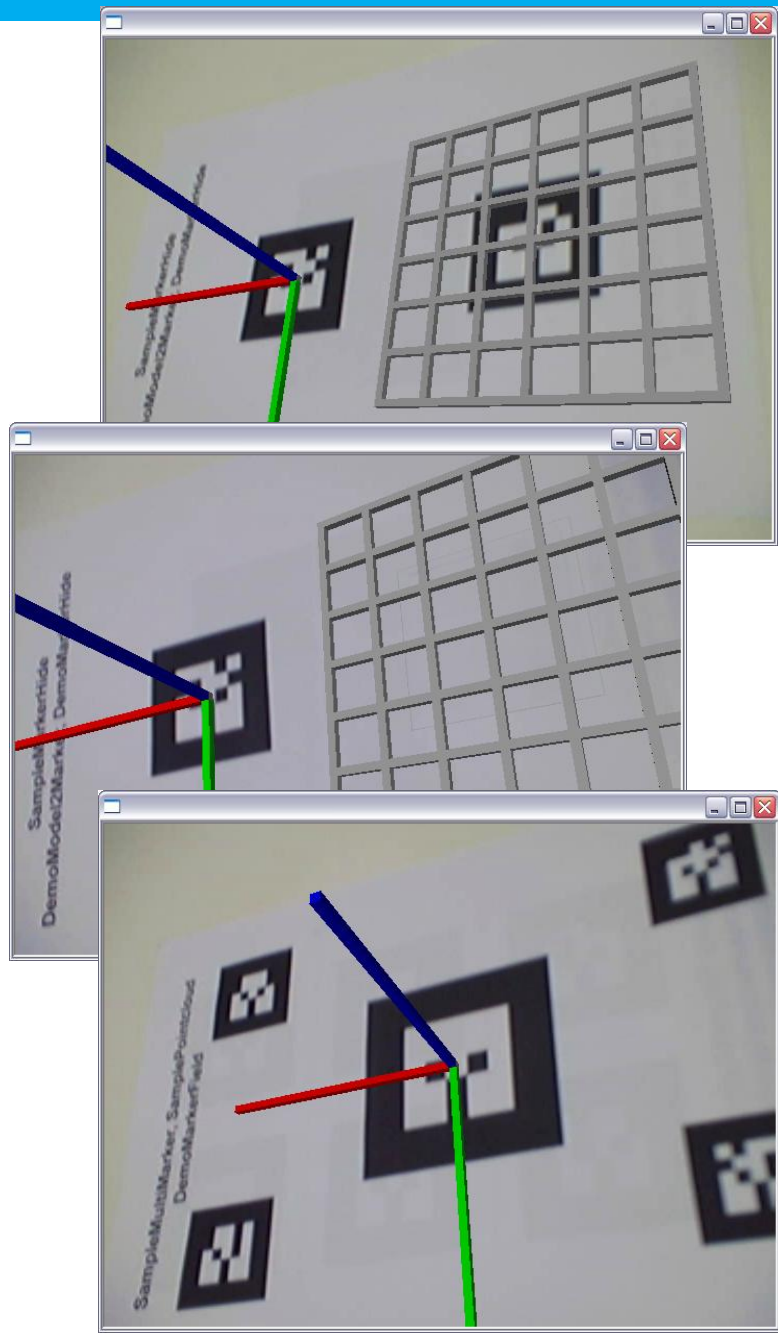
Demo programs: core + advanced

- There are three demo programs that demonstrate the core ALVAR features, and 3 for the advanced features.
- The core demos use only HighGui for acquisition and refer to hard-coded data files (no command-line parameters).
- The advanced demos use all installed acquisition plug-ins, and while they also refer to hard-coded data files, some settings can be overridden with command-line parameters.
- The command-line parameters of the advanced demos are: webcam index, camera calibration file, and app-specific option.



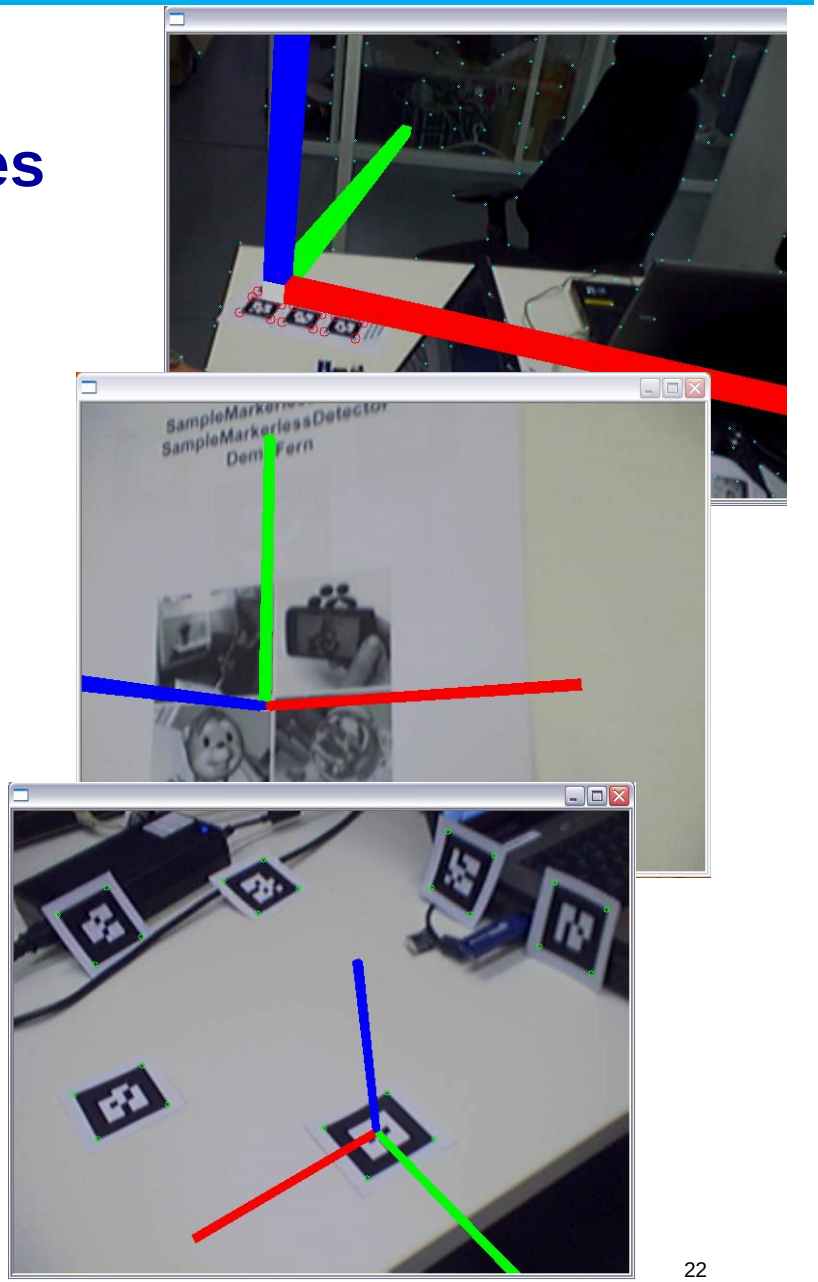
Demo programs, core features

- **Model2Marker** – Identifies two independent markers and renders simple OSG model on top of both markers.
- **MarkerHide** – Identifies two independent markers, renders simple OSG model on top of both markers and hides one of the markers using ALVAR hide-texture generation.
- **MarkerField** – Identifies a set of markers in predefined configuration and renders a single OSG model as long as any one of the markers is visible.



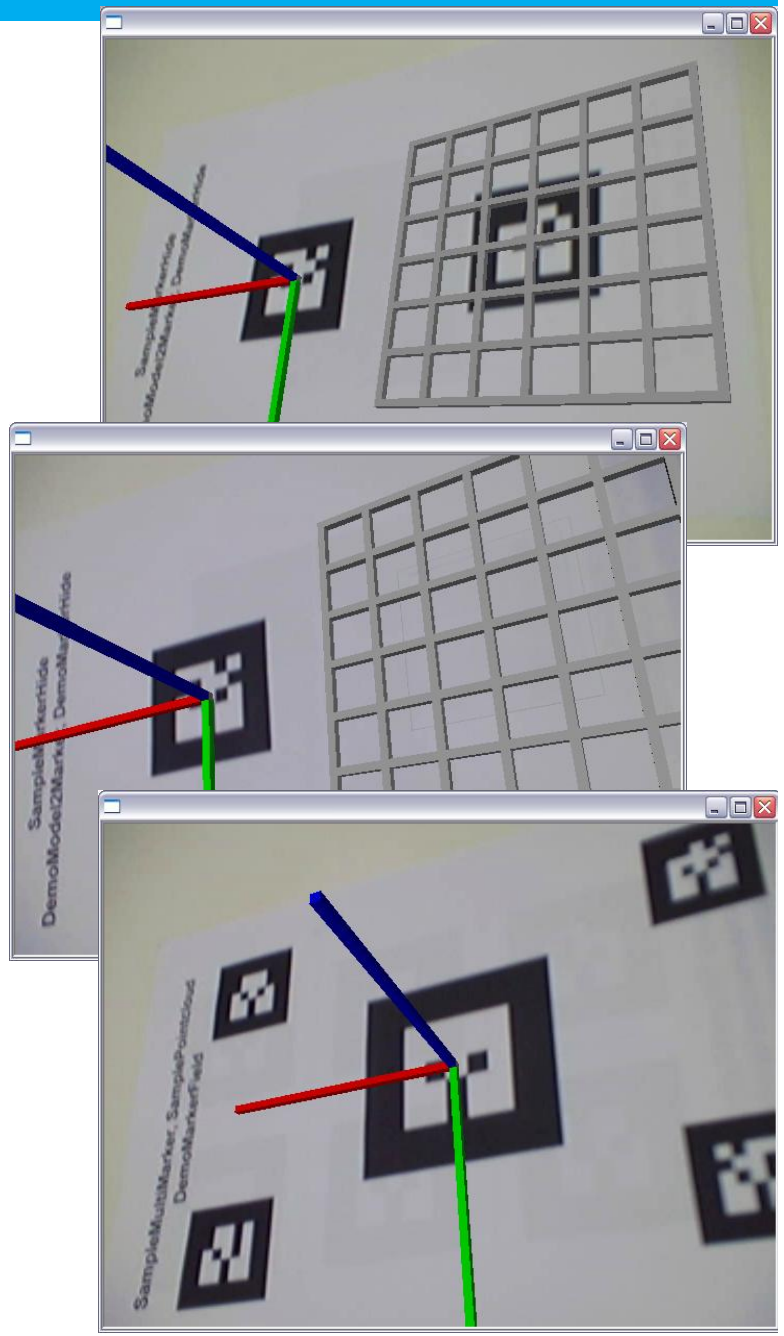
Demo programs, advanced features

- **OsgSfM** – Identifies a predefined marker configuration and renders an OSG model. Reconstructs features with two alternative methods for tracking in the surroundings.
- **OsgFern** – Uses an image as a marker and renders an OSG model. Uses the Fern's algorithm.
- **Osg3DMarkerField** – Deduces the spatial configuration of a 3D marker field, then renders an OSG model on top of the selected marker in this 3D marker field.



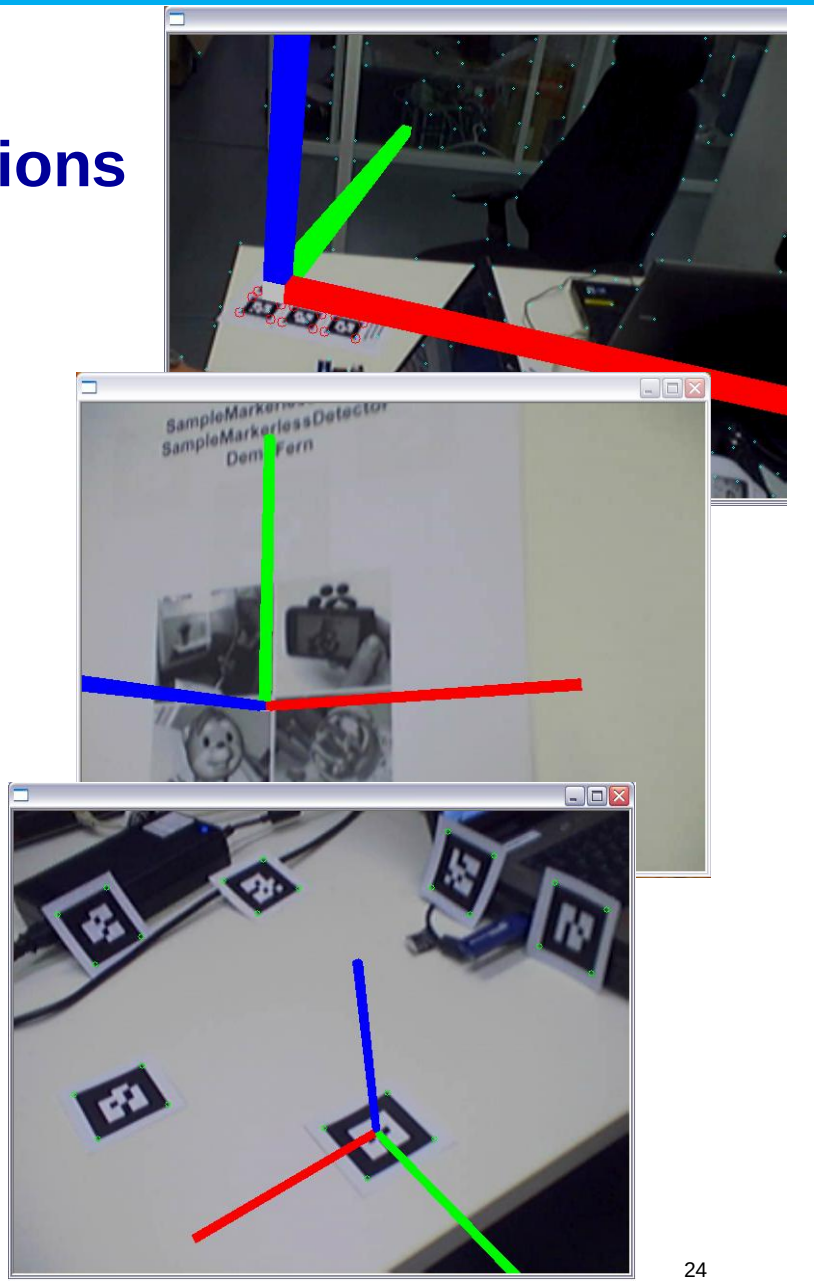
Demo programs (core): Instructions

- **Model2Marker** – Print markers 5 and 10 in the ‘Alvar.pdf’ file in the **doc** folder and show them to your webcam.
- **MarkerHide** – Print markers 5 and 10 in the ‘Alvar.pdf’ file and show them to your webcam.
- **MarkerField** – Print multimarker in the ‘Alvar.pdf’ file and show it to your webcam.
- With all three programs, if file ‘calib.xml’ is found, it is used as your camera calibration. This file can be created with the utility program ‘SampleCamCalib.exe’.



Demo programs (advanced): Instructions

- **OsgSfM** – The default multimarker to be tracked is in file ‘mmarker.xml’ (3rd cmd-line param overrides), which refers to multimarker in the ‘Alvar.pdf’ file. Some keyboard commands are available, see the command window.
- **OsgFern** – The default image file to be tracked is in ‘AlvarSlide.jpg’. Use 3rd cmd-line parameter to enter some other image to be used as a marker, its size should be about 200x200 pixels.
- **Osg3DMarkerField** – Create a 3D marker field using ALVAR markers 0 - 11. Marker 0 is required (its pose is reported as the marker field pose), others are optional. Use 3rd cmd-line parameter to use another pose marker.
- See **doc** and **data** folders to find these files.



ALVAR Markers to Print

- The **doc** folder contains the 'Alvar.pdf' file with markers to print
- Print the markers you intend to use, attach the prints to a level surface
- Each marker contains a numerical ID, which must be entered as the marker ID in the program
- Marker IDs 5 and 10 are used by some demo programs
- The camera calibration (see next slides) uses the chessboard pattern
- You can find a multimarker (i.e. 2D marker field) to print in the 'Alvar.pdf' file, and the corresponding XML file is 'mmarker.xml' in the **data** folder, to be used with the ALVAR multimarker feature
- You can create image files (*.png) with more markers with the utility program 'SampleMarkerCreator.exe' (see next slides)

ALVAR Utility Programs (1/5)

- Two of the sample programs are also useful utility programs: SampleCamCalib.exe and SampleMarkerCreator.exe
- In the separately downloadable **Bin** ALVAR distribution package you'll find these two ALVAR utilities
- SampleCamCalib.exe is used to calibrate your webcam, using the chessboard pattern
- SampleMarkerCreator.exe is used to create marker image files which you can print. It can also create multimarkers, i.e. 2D marker fields

ALVAR Utility Programs (2/5)

SampleCamCalib

```
ALVAR 2.0.0 - A Library for Virtual and Augmented Reality
Copyright 2007-2012 VTT Technical Research Centre of Finland
Licensed under the GNU Lesser General Public License
Built on 2012-05-29 for Windows 6.1 x86

SampleCamCalib
==============

Description:
  This is an example of how to use the 'Camera' and 'ProjPoints' classes
  to perform camera calibration. Point the camera to the chessboard
  calibration pattern (see ALVAR.pdf) from several directions until 50
  calibration images are collected. A 'calib.xml' file that contains the
  internal parameters of the camera is generated and can be used by other
  applications that require a calibrated camera.

Usage:
  samplecamcalib.exe [device]

    device     integer selecting device from enumeration list (default 0)
               highgui capture devices are prefered

Keyboard Shortcuts:
  q: quit

Available Plugins: file, highgui

Enumerated Capture Devices:
* 0: highgui_0
```

SampleMarkerCreator

```
ALVAR 2.0.0 - A Library for Virtual and Augmented Reality
Copyright 2007-2012 VTT Technical Research Centre of Finland
Licensed under the GNU Lesser General Public License
Built on 2012-05-29 for Windows 6.1 x86

SampleMarkerCreator
===================

Description:
  This is an example of how to use the 'MarkerData' and 'MarkerArtoolkit'
  classes to generate marker images. This application can be used to
  generate markers and multimarker setups that can be used with
  SampleMarkerDetector and SampleMultiMarker.

Usage:
  samplemarkercreator.exe [options] argument

    65535              marker with number 65535
    -f 65535           force hamming(8,4) encoding
    -1 "hello world"   marker with string
    -2 catalog.xml     marker with file reference
    -3 www.vtt.fi      marker with URL
    -u 96              use units corresponding to 1.0 unit per 96 pixels
    -uin               use inches as units (assuming 96 dpi)
    -ucm               use cm's as units (assuming 96 dpi) (default)
    -s 5.0             use marker size 5.0x5.0 units (default 9.0x9.0)
    -r 5               marker content resolution -- 0 uses default
    -m 2.0             marker margin resolution -- 0 uses default
    -a                 use ArToolkit style matrix markers
    -p                 prompt marker placements interactively from the user


Prompt marker placements interactively
  units: 1 cm 0.393701 inches
  marker side: 9 units
  marker id (use -1 to end) [0]:
```

ALVAR Utility Programs (3/5)

- SampleCamCalib.exe is used to calibrate your webcam, using the chessboard pattern (the *.pdf file in the **doc** folder)
- Print the slide containing the chessboard pattern. Attach the print to a level surface
- Start the program, and show the chessboard pattern to your webcam in different positions, distances, and orientations
- The program acquires 50 samples, about once a second. When a sample is acquired (the chessboard pattern is visible), the found pattern is shortly shown in red
- Calibration result is written to file 'calib.xml'
- By default, SampleCamCalib.exe calibrates the default camera using the default resolution. However, if either of these is not the one you intend to use, enter command-line argument "1" for your first camera (resolution and other options will be prompted), "2" for your 2nd camera, etc.

ALVAR Utility Programs (4/5)

- SampleMarkerCreator.exe is used to create marker image files which you can print. It can also create multimarkers, i.e. 2D marker fields
- To create an image file containing a single marker, enter the marker number (=ID) as a command-line parameter
- If no command-line parameters are entered, the program goes to the interactive mode; in this mode you can enter the IDs and relative positions of several markers, creating a multimarker. Defaults are sensible, so you can just keep pressing <Enter> until your multimarker is big enough
- The output image of the program is written into file 'markerdata_*0*.png' (where *0* is replaced with the marker id). Default marker size is 9x9 cm
- The output image of a multimarker employing e.g. markers 0, 1, 2, and 3 is written into file 'markerdata_*0_1_2_3*.png', whereas the XML definition of this multimarker is written into 'markerdata_*0_1_2_3*.xml' (very long file names are truncated)

ALVAR Utility Programs (5/5)

SampleMarkerCreator.exe usage options and additional instructions

- The program prompts for marker id and its position in the used units (3 questions per marker). The default unit is 'cm' and the default marker size is 9x9 cm
- If the user just presses <Enter> to all prompts, the program creates a square marker field which keeps growing (2x2 => … => 3x3 => … => 4x4 etc.)
- When enough markers have been added to the marker field, the user should enter -1 for the ID of the next marker, which ends the prompting

- If the user wishes to use other units or a different marker size, these can be entered as command-line parameters before the interactive mode, e.g.:
- > samplemarkercreator.exe -uin -s 1.0 –p

- If the user wishes to enter a marker field design into a separate text file, it is also possible (contents of 'test.txt' is shown to the right):
- > samplemarkercreator.exe -p < test.txt

- If the user wishes to use markers of different sizes, the data for all the markers must be entered as command-line parameters:
- > samplemarkercreator.exe -s 18 -xy -36 0 255 -xy -36 -36 254 -xy 0 -36 253 -s 9 –p
- This example sets 18x18 cm markers 255, 254 and 253 to the left, up-left, and up from the origin. Then the program enters the interactive mode to prompt for additional 9x9 cm markers

File 'test.txt' contains the numbers that the user usually enters manually, e.g.:

```
0
-5
-10
1
10
-5
2
5
10
3
-10
5
-1
```

ALVAR Directories

- **bin** - The compiled binaries will appear in a subdirectory matching the selected build subdirectory
- **build** - The building environment is in a matching subdirectory. See file doc/compiling.txt
- **data** – contains data files used by some samples and demos
- **demo** – OpenSceneGraph programs that demonstrate how to use the library with 3D graphics
- **doc** – HTML Documentation. Generated using Doxygen (e.g. "make doc"). *Also contains sample markers in the* 'Alvar.pdf' *file.*
- **include** – contains the ALVAR C++ header files (**Sdk** package only)
- **sample** - Samples that demonstrate how to use the library.
- **src** - Sources for the ALVAR library (**Src** package only). Note that Alvar.h is different from the others; it is generated separately for each build environment based on Alvar.h.cmake

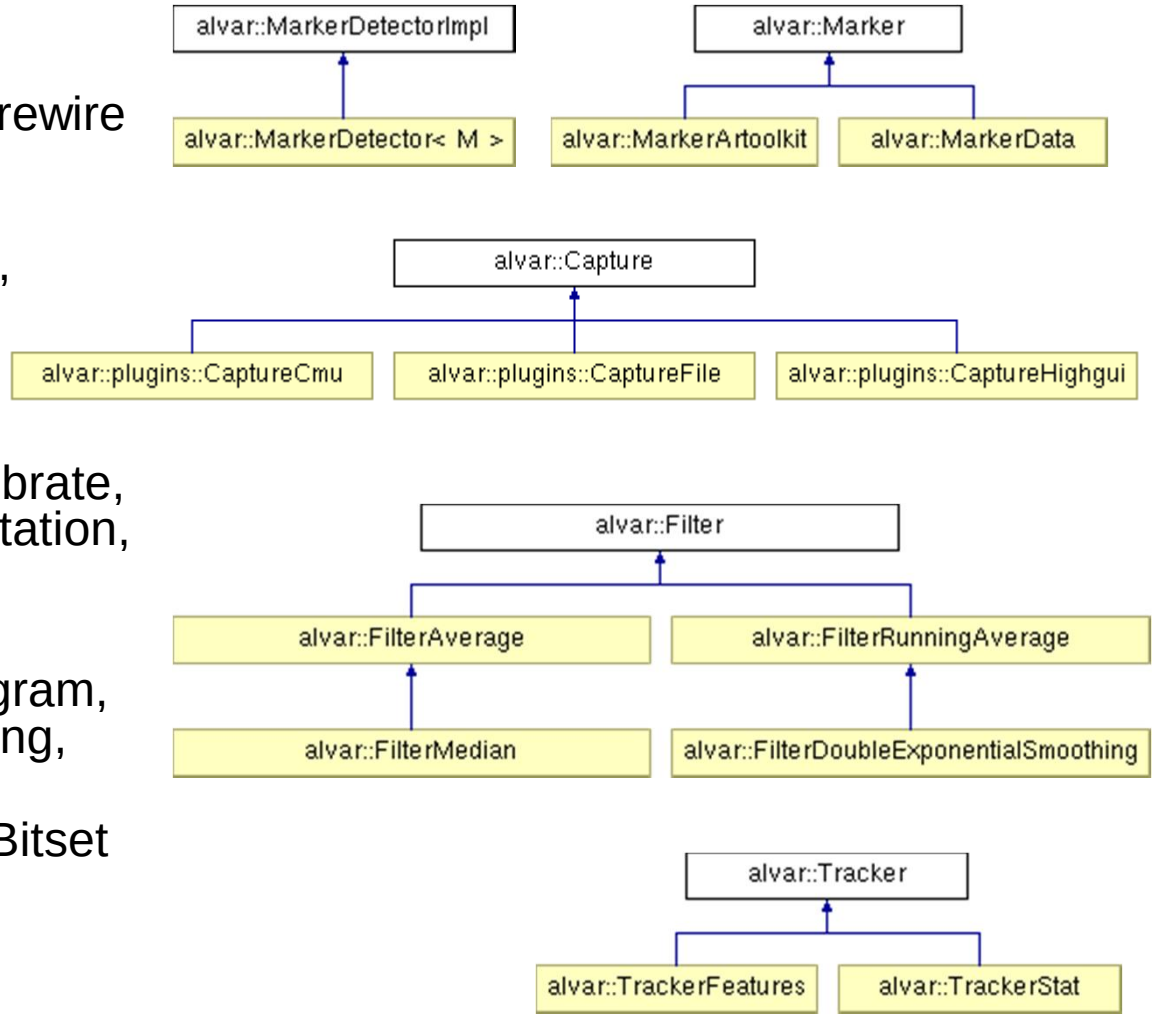
Basic ALVAR Usage (Sdk package), Compiling ALVAR Applications

- The file ‘compiling.txt’ in the **doc** folder contains the most up-to-date instructions on what 3rd-party libraries to download and how to compile AR applications (e.g. ALVAR Samples and Demos) using the ALVAR library

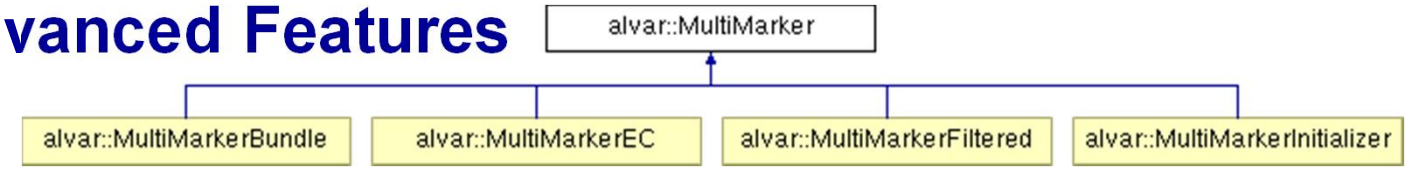
ALVAR: Main Core Features

- Capture video from USB camera, Firewire camera or AVI file (using plugins).
- Detecting Markers and predefined MultiMarkers. Marker types: ALVAR, ARToolkit, custom
- Filters for data sequences
- Tracking image features
- Camera/Homography methods: Calibrate, Distort, Undistort, CalcExteriorOrientation, ProjectPoints

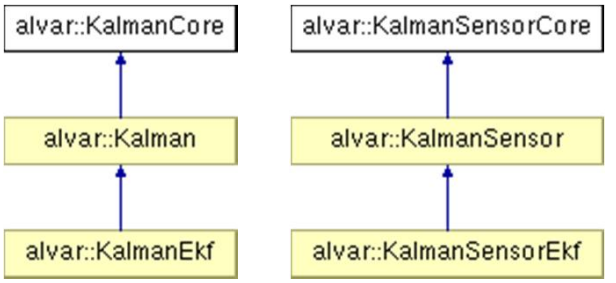
- Further utils: Threads, Mutex, Histogram, Serialization, Image Labeling, Drawing, HideTexture, …
- Types: Point, Line, Rotation, Pose, Bitset

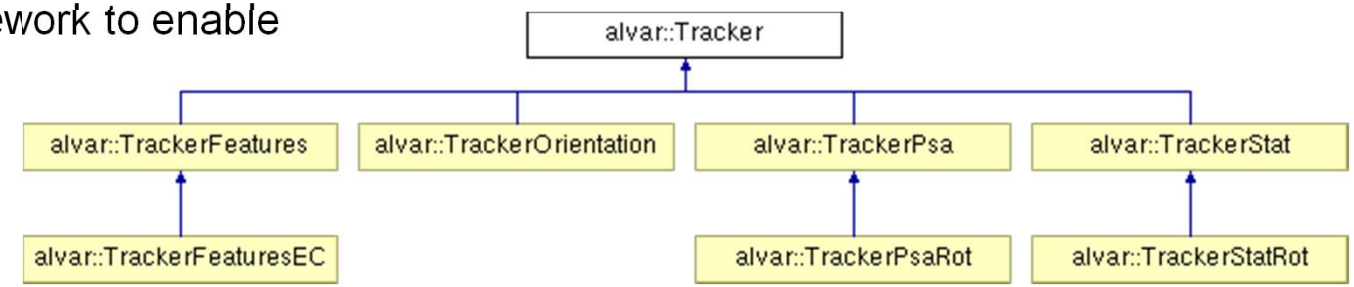


ALVAR: Main Advanced Features



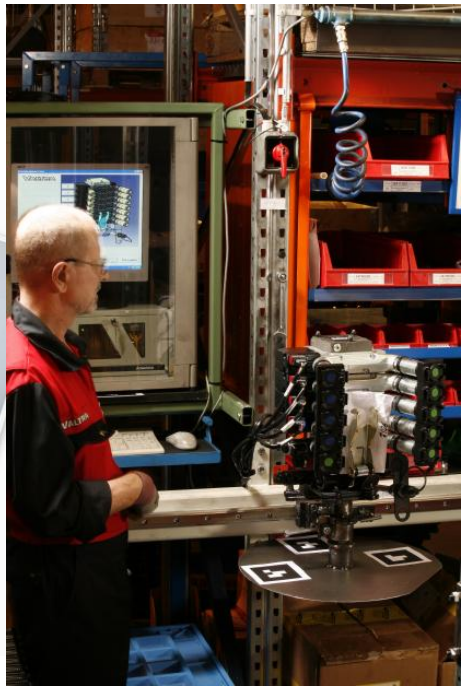
- Methods to deduce/optimize MultiMarker setups
- SimpleSfM: Structure for motion to use features in addition to markers. Pose update optimization.
- External container versions of several methods
- Non-linear optimization using Gauss-Newton, Levenberg-Marquardt and Tukey m-estimator
- Kalman filter, EKF, Unscented Kalman filter
- More methods for tracking image features
- Further utils: Container3d, Ransac, TrifocalTensor, IntegralImage, IntegralGradient, …
- Fern's classification framework to enable markerless tracking





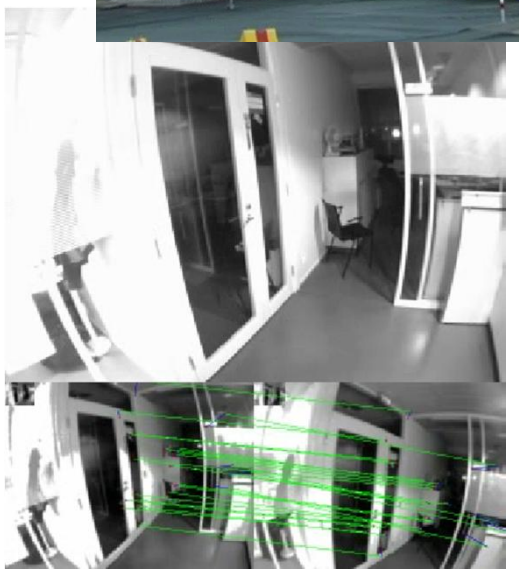
Some Application Examples

- Print media
- Interior design
- Building & construction
- Augmented assembly
- Indoors locationing
- Virtual worlds



Other Demonstrations of the AR Team (not yet in ALVAR)

- 3D-model based tracking
- Image database e.g. for tracking init/recovery
- Photorealistic rendering
- Plugin interface for external sensors (e.g. inertial measurement unit)



Thank You!

Petri Honkamaa

Main contact:
Prof. Charles Woodward
VTT Technical Research Centre of Finland
Vuorimiehentie 3, Espoo, Finland
Tel: +358 20 722 5629
charles.woodward@vtt.fi

http://www.vtt.fi/multimedia