



Alvar

A Library for Virtual and Augmented Reality
Copyright © 2009 VTT Technical Research Centre of Finland



Business from technology

Introduction

Alvar is a library for making virtual- and augmented reality applications. The first versions of the library mainly assists in the development of marker-based augmented reality applications. Future versions will not only provide tools for creating markerless augmented reality solutions, but also allow the development of virtual reality applications.

The Alvar library is designed to be very flexible. It currently requires only one 3rd party library (OpenCV) for all the main functionality. Alvar offers high-level tools and methods for the rapid prototyping of augmented reality solutions 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 create their own solutions using alternative approaches or completely new elements and algorithms.

Requirements

Alvar has been tested with the following environments:

- Windows XP 32-bit, Microsoft Visual Studio 7 and 8.

Alvar core library requires the following 3rd party library:

- OpenCV 1.0

Alvar UI library and samples require the following:

- GLUT 3.7
- CMake 2.6

Basic Usage

- Ensure that you have a suitable development environment. Currently the library has been used with the following.
 - Microsoft Visual Studio 7 and 8
 - MinGW + GNU Make
- Install the required 3rd party libraries: OpenCV, GLUT (needed only if you use also 'alvarui' library).
- Install the Alvar library.
- Develop your application. Include the needed Alvar headers in your source directory and link to the Alvar library matching your development environment (e.g. bin/msvc80/alvar.lib).
- Copy the OpenCV (and GLUT) runtime libraries where your application can find them (e.g. the exe-directory).

Compiling the Samples

- Ensure that you have a suitable development environment.
- Install the required 3rd party libraries: OpenCV, GLUT, CMake, Doxygen.
- Install the Alvar library.
- Note that the CMake binary directory must be in your system's PATH environment variable.
- Generate the development environment to build the samples by running the generate-script of your choice (e.g. build/msvc80/generate.bat).
- Fill in the missing information for CMake (e.g. GLUT_ROOT_PATH).
- Do NOT change the 'where to build binaries' directory.
- Use the generated development environment to compile the samples.

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 Building.
- **doc** - Documentation. Generated using Doxygen (e.g. "make doc").
- **sample** - Samples that demonstrate how to use the library.
- **src** - Sources for the Alvar library (alvar). Note that Alvar.h is different from the others; it is generated separately for each build environment based on Alvar.h.in . [In binary distributions this directory only contains only the header files.]
- **src/ui** - Sources for the Alvar UI library (alvarui). [In binary distributions this directory contains only the header files.]

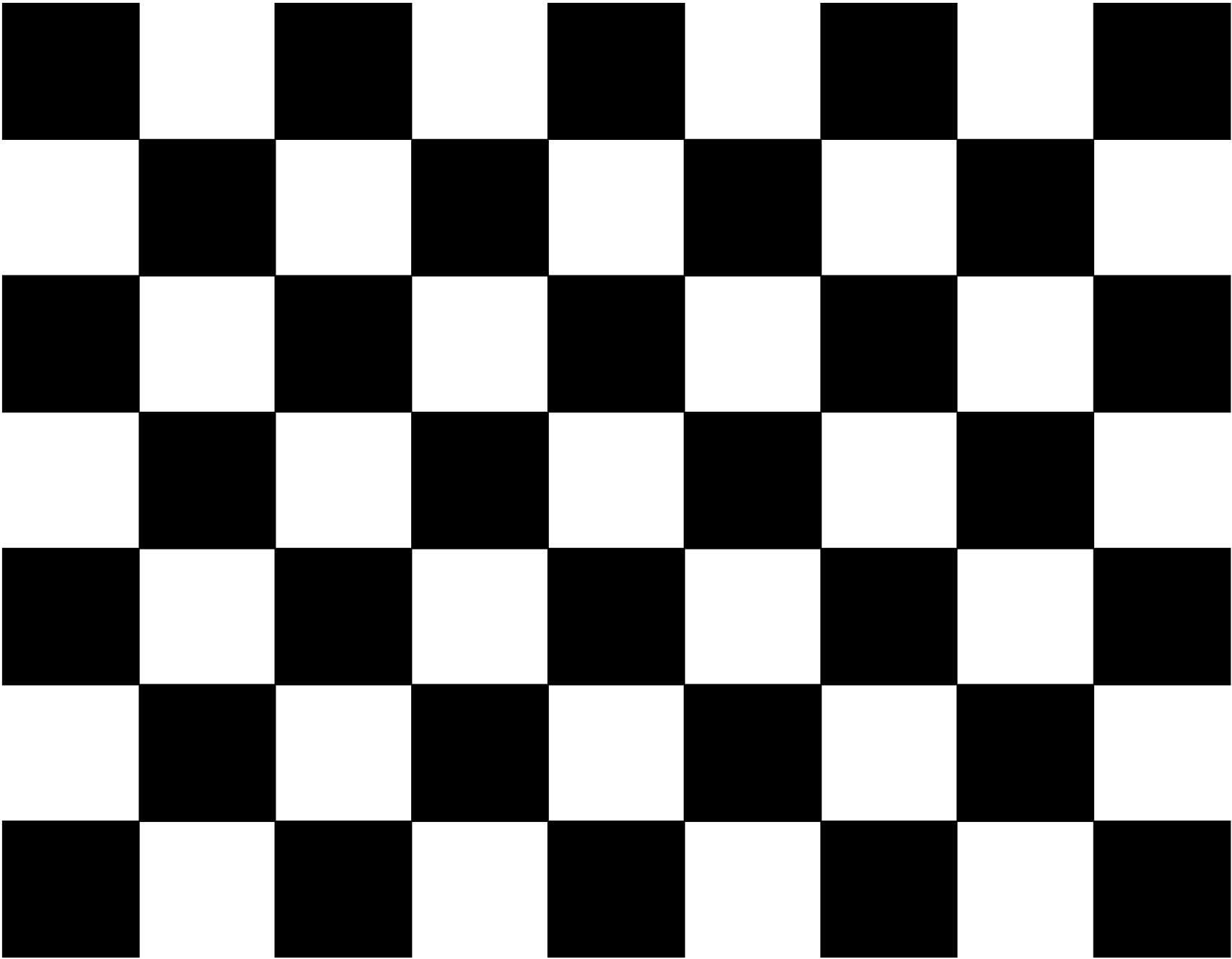
Main Features

- Detecting and tracking 2D markers.
 - Currently two types of square matrix markers are supported. Future marker types can easily be added.
 - Accurate marker pose estimation.
 - Tracking heuristics to identify markers that are "too far" and to recover from occlusions for example.
- Using setup of multiple markers for pose detection.
 - The marker setup coordinates can be set manually, or
 - They can be automatically deduced using various methods.
- Tools for calibrating cameras. Distorting/undistorting points, projecting points and finding exterior orientation using point-sets.
- Several basic filters a Kalman library (e.g. for sensor fusion).
- Several methods for tracking using optical flow.
- Building and showing "Hide"-textures (e.g. for hiding textures).

Example programs

Alvar library also includes several example programs that demonstrate how to use the toolkit (with source code).

- **SampleCamCalib** Camera calibration with chessboard calibration pattern.
- **SampleCamCalibMarker** Camera calibration with detected markers.
- **SampleCvTestbed** Example of using the *CvTestbed* class. *CvTestbed* is a class for making quick OpenCV prototype applications.
- **SampleFilter** Example of using the various filters: *FilterAverage*, *FilterMedian*, *FilterRunningAverage*, *FilterDoubleExponentialSmoothing* and *Kalman*.
- **SampleLabeling** Example of how to label images.
- **SampleMarkerCreator** Generating and saving a marker images.
- **SampleMarkerDetector** Detecting *MarkerData* markers and visualizing them using *GlutViewer*. *GlutViewer* is a simple OpenGL interface using GLUT.
- **SampleMultiMarker** Using preconfigured *MultiMarker* setup.
- **SampleMultiMarkerBundle** Using various methods to automatically deduce and optimize *MultiMarker*-setups.
- **SampleTrack** Tracking the optical flow using various methods.



SampleCamCalib

SampleMultiMarker

