Alvar
A Library for Virtual and Augmented Reality

General Overview

Augmented Reality Team
VTT Technical Research Centre of Finland

Introduction

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 first versions of the library mainly support marker-based augmented reality applications. Future versions will include tools for markerless augmented reality, as well as photorealistic visualization and virtual reality.

Introduction

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 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. The example programs use GLUT and OpenSceneGraph.

Features

- 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.

Features

- 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.

Platforms

Alvar is officially supported and tested on the following platforms.

- Windows XP 32-bit, Microsoft Visual Studio 2003 (7.1), 2005 (8.0) and 2008 (9.0)
- Linux 32-bit, GCC 4.4
- Linux 64-bit, GCC 4.4

Dependencies

The Alvar library depends on the following libraries.

- OpenCV 1.0

The Alvar samples depend on the following libraries and tools.

- GLUT 3.7
- CMake 2.6

Usage

Please see the instructions in doc/Compiling.txt for more information.

Samples

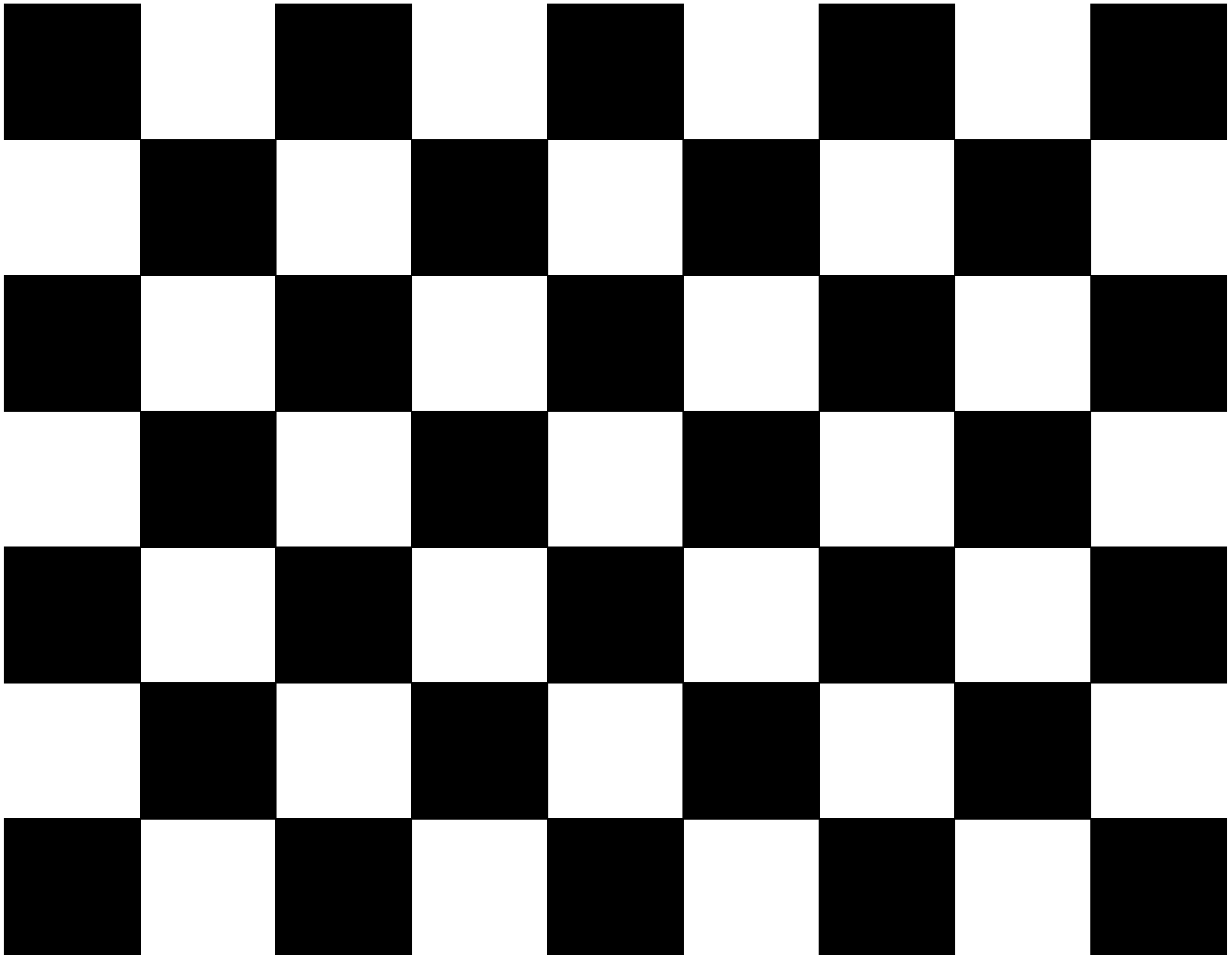
- *SampleCamCalib* – This is an example of how to use ProjPoints and Camera classes to perform camera calibration using a chessboard pattern.
- *SampleCvTestbed* – This is an example of how to use the CvTestbed and Capture classes in order to make quick OpenCV prototype applications.
- *SampleFilter*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

Samples

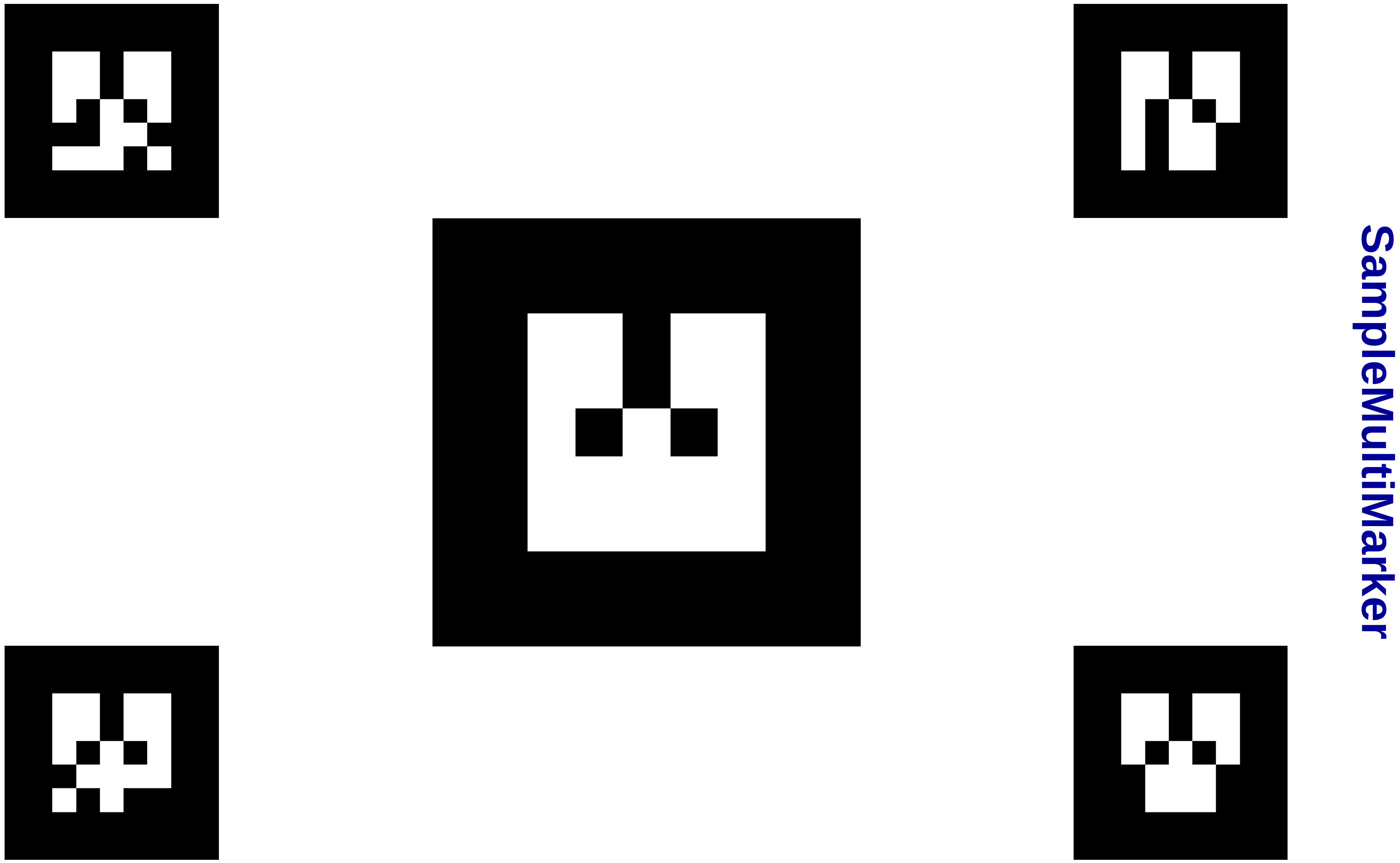
- *SampleLabeling* – This is an example of how to label images using LabelImage and MarchEdge.
- *SampleMarkerCreator* – This is an example that demonstrates the generation of MarkerData markers and saving the image using SaveMarkerImage.
- *SampleMarkerDetector* – This is an example that shows how to 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.
- *SampleMultiMarker* – This is an example that demonstrates the use of a preconfigured MultiMarker setup.

Samples

- *SampleMultiMarkerBundle* – This is an example that automatically recognising MultiMarker setups using MultiMarkerFiltered and optimizes it with 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 or TrackerStat.



SampleCamCalib


SampleMultiMarker

SampleMarkerDetector
SampleMultiMarkerBundle