PA198
Augmented Reality Interfaces

Lecture 3

Augmented Reality Software

Fotis Liarokapis
liarokap@fi.muni.cz

03rd October 2016

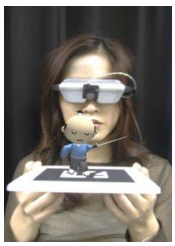
AR Solutions

- In the past only few tools
  - Had to do a lot of coding…
- Nowadays loads of solutions
  - Open Source
  - Proprietary
  - End-to-End Branded App Solutions

Open Source

ARToolKit

- Dual-license:
  - GPL, commercial
- C-library
- Ported to different languages and platforms
  - i.e. Android, Flash, Silverlight, etc
- More later on…



Argon

- AR browser by Georgia Tech's GVU Center
- Uses a mix of KML and HTML/JavaScript/CSS for developing AR applications
- Any web content can be converted into AR content
  - With appropriate meta-data and properly formatted
- As of November 2011, available for iPhone only

http://argon.gatech.edu/

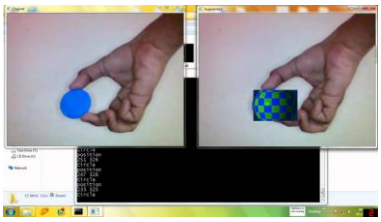
ArUco

- Minimal library for AR applications
  - Trivial integration with OpenGL and OGRE
- OpenCV implementation
  - Up to 1024 different markers
- Licenses:
  - BSD, Linux, Windows

http://www.uco.es/investiga/grupos/ava/node/26

JavaCV

- Java/Android interface to OpenCV
- Licenses
  - GPLv2



https://code.google.com/p/javacv/wiki/OtherResources

ATOMIC Authoring Tool

- Created as a front end (Graphic Interface) for use ARToolKit without having to know programming
  - Written in Processing
- Multi-platform authoring for creating AR applications
  - Microsoft Windows, Linux and Mac OS X operating systems

http://sourceforge.net/projects/atomic-project/

Goblin XNA

- Platform for researching 3D user interfaces emphasizing games
  - Mobile augmented reality and virtual reality
- Implementation in C#
- Based on Microsoft XNA Game Studio 4.0
- BSD license

https://goblinxna.codeplex.com/

GRATF

- GRAFT (Glyph Recognition And Tracking Framework)
- Does localization, recognition and pose estimation of optical glyphs in still images and video files

http://www.aforgenet.com/projects/gratf/

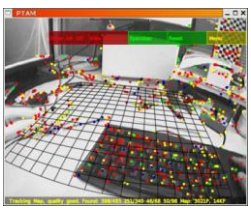
Mixare

- Mixare (mix Augmented Reality Engine) AR browser
  - Published under the GPLv3
  - Available for Android iPhone

http://www.mixare.org/

PTAM

- PTAM (Parallel Tracking and Mapping) is a camera tracking system for AR
- Requires no markers, pre-made maps, known templates or inertial sensors
- Non-commercial use only



http://www.robots.ox.ac.uk/~gk/PTAM/

DroidAR

- AR framework for Android
- Features location based and marker based AR
- Open source (dual-license):
  - GPLv3 or commercial

https://github.com/bitstars/droidar

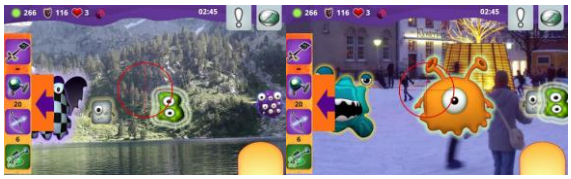
GeoAR

- Open source (Apache 2.0 License) browser for Android
- Features location based AR and a flexible data source framework

https://wiki.52north.org/bin/view/Projects/GeoAR

BeyondAR

- Open source (Apache 2.0 License)
- AR framework based on geo-localisation for Android



http://beyondar.com/

Proprietary

Kudan AR Engine

- AR SDK for iOS and Android devices
- Powerful Rendering
  - Multi-million polygon 3D models
- Advanced Tracking
  - Markerless

https://www.kudan.eu/

Vuforia Augmented Reality SDK

- The Vuforia SDK supports different types of targets
  - both 2D and 3D, including multi-target configurations, cylinder targets to track images on a cylindrical surface, marker less image targets, frame markers and cloud recognition targets to track 1 million targets simultaneously
- SDK supports both iOS and Android

https://www.qualcomm.com/products/vuforia

Vuforia Features

- Provide faster local detection of targets with capacity of tracking 5 targets simultaneously
- Efficient tracking in low light conditions and even though target is covered partially
- Extended tracking capabilities, which enable app to keep tracking targets and helps maintain a consistent reference for augmentations of object even when the targets are no longer visible in real time camera view

https://www.qualcomm.com/products/vuforia

Vuforia Video

https://www.youtube.com/watch?v=GrP_xwrCtB0

Metaio SDK

- Modular framework which consists of different components
  - Rendering, capturing, tracking and the sensor interface
- Compatible with all major platforms:
  - Android, IOS, Unity3D and Windows
- SDK contains:
  - Marker or marker-less 2D and 3D tracking
  - POI's tracking
  - Support for QR code and barcode reading
  - Built in 3D renderer
  - Optimizations for mobile chips
  - etc.

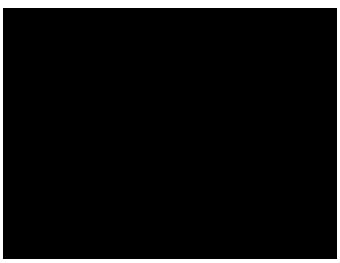
https://www.metaio.com/

Metaio Features

- Provides high level of abstraction
- Provides powerful 3D rendering engine and highly advance tracking
- Support for obj, fbx and md2 model formats for 3D objects
- Direct loading of 3D models is supported with abstraction of OpenGL calls
- Includes AREL scripting language
  - AR experiences based on common web technologies
    - i.e. XML, HTML5 and JavaScript
- Bought by Apple

https://www.metaio.com/

Metaio Video



https://www.youtube.com/watch?v=2HRW-yDgzA8

End-to-End Branded App Solutions

Alive app

- Developed by Times Internet Ltd
- Recognise pre-trained images and overlay an image, video or 3D content
- Available on:
  - Android, iOS, Windows, Blackberry Symbian and Java

http://www.alivear.com/

Aurasma

- HP Autonomy's AR platform
- Available as SDK or free app for iOS and Android
- Recognize real world images and then overlay graphics
- Created in Cambridge

https://www.aurasma.com/

Blippar

- Visual browsing app using image-recognition and AR technologies
- Focused on smartphones, tablets or wearables

https://blippar.com/en/

Junaio

- AR browser designed for 3G and 4G mobile devices
- First AR browser that has overcome the accuracy limitations of GPS through LLA Markers
  - i.e. latitude, longitude, altitude marker, patent pending

http://www.junaio.com/

XARMEX

- XARMEX (eXtended Augmented Reality for Military EXercise)
- AR-aided Close quarters combat simulation system, combining motion detection hardware with image overlay/stabilization software to create realistic military simulation environments and computer games

https://en.wikipedia.org/wiki/XARMEX

Layar

- Mobile browser
- Can enhance
  - flyers, postcards, packaging or any other item with interactive content, including video messages, Web and social links, photo slideshows

https://www.layar.com/

Wikitude

- Mobile AR technology provider
  - Based in Salzburg, Austria
  - Founded in 2008
- Initially focused on providing location-based AR experiences through the Wikitude World Browser App
- Wikitude SDK (2012)
  - Uses image recognition and tracking
  - Geo-location technologies
  - Free plug-in for Unity

https://www.wikitude.com/

Wikitude .

- Content in the Wikitude World Browser is mostly user generated
- Content can be added by a web interface
  - i.e. KML and ARML
- Web services are available to register the delivery of dynamic data
- Wikitude is a W3C member and OGC member and is working to develop ARML further as part of a W3C ARML project

https://www.wikitude.com/

Wikitude Video

https://www.youtube.com/watch?v=JnemoNQTa_w

D'Fusion

- Integrates real time interactive 3D graphics contents on a live video stream
- SDK available in different platforms
  - i.e. desktop, mobile and special Flash Plug-in

http://www.t-immersion.com/products/dfusion-suite

D'Fusion Features

- It is more UI based
  - D'Fusion Studio
  - D'Fusion CV
- Enables to build the whole scenario through the GUI
- One bundle scenario will work on both Android & iPhone
- Supports multi-tag and face-tracking

http://www.t-immersion.com/products/dfusion-suite

D'Fusion Video

https://www.youtube.com/watch?v=6NKT6eUGJDE

ARmedia 3D

- Uses a 3D model tracking approach
  - Recognizes planar images and complex 3D objects independently of their size and geometry
- SDK architecture consist of renderer to render 3D model, tracker to track the target, capture for capturing frames from the device camera and interface to native android and iOS
- Cross-platform and implemented in C/C++

http://www.armedia.it/index.php

ARmedia Features

- Provides 3D tracking of real world objects in real time in changing lighting conditions
- Provides modularity
  - Can integrate between different tracking algorithms and different 3D engines and interface
  - Different tracker parameters can be fixed to fine tune the results
  - Easily integrate with any other AR platform
  - 3D target creation and management services are available on cloud

AR Media Video

https://www.youtube.com/watch?v=g4_G6VtCssI

ARToolKitPlus

- Mobile version of ARToolKit
  - First approach
  - Development stopped June 2006
- Succeeded by Studierstube Tracker
  - It is a marker tracking library
    - http://handheldar.icg.tugraz.at/stbtracker.php

http://handheldar.icg.tugraz.at/artoolkitplus.php

Comparisons

Comparison of AR SDK's

- Based on free and open-source SDK's
- Popular SDK's
  - Metaio
  - Vuforia
  - Wikitude
  - D'Fusion
  - Armedia
  - ARToolKit

Based on License Type

AR SDK Type		Vuforia	Metaio	Wikitude	ARToolKit	D'Fusion	ARmedia
License	Open source	×	×	×	✓	×	×
	Free	✓	✓	✓	✓	✓	✓
	Commercial	✓	✓	✓	✓	✓	✓

Amin, D., Govilkar, S. Comparative Study of Augmented Reality SDK's, Int'l Journal on Computational Sciences & Applications, 5(1): 11-26, February 2015.

Based on Platform Supported

AR SDK Type		Vuforia	Metaio	Wikitude	ARToolKit	D'Fusion	ARmedia
License	iOS	×	✓	×	✓	✓	✓
	Android	✓	✓	✓	✓	✓	✓
	Window	✓	✓	✓	✓	✓	✓

Amin, D., Govilkar, S. Comparative Study of Augmented Reality SDK's, Int'l Journal on Computational Sciences & Applications, 5(1): 11-26, February 2015.

Based on Marker Generation

AR SDK Type	Vuforia	Metaio	Wikitude	ARToolKit	D'Fusion	ARmedia
Marker generation	Online target manger. Support generation of frame, image markers.	No online tool. Provides readymade 512 different ID markers	Online target manager tool. Provide creation of target collection of multiple targets.	Online tool to create marker. Provide set of predefined square markers in a PDF file.	D'Fusion studio. Provide dataset of 500 images to use as marker.	Provide plug-in for Google SketchUp. Provides set of predefined markers.

Amin, D., Govilkar, S. Comparative Study of Augmented Reality SDK's, Int'l Journal on Computational Sciences & Applications, 5(1): 11-26, February 2015.

Based on Tracking

AR SDK Type		Vuforia	Metaio	Wikitude	ARToolKit	D'Fusion	ARmedia
Tracking	Marker	Frame markers, image target, text targets.	ID, picture and LLA marker, QR and Barcode	Image, barcode tracking	Square marker, multiple marker tracking	Multiple marker tracking ,Barcode tracking	Track Fiducial marker.
	GPS	×	✓	✓	×	✓	✓
	IMU	×	✓	✓	×	✓	✓
	Face	×	✓	×	×	✓	×
	Natural Feature	✓	✓	✓	✓	✓	✓
	3D object	✓	✓	×	×	✓	✓
	Others	Extended tracking, Localized Occlusion detection	Instant 3D maps tracking ,3D SLAM ,extended image tracking	Hybrid tracking, extended tracking	6D marker tracking(real-time planar detection)	Recognized interactive zone through finger tracking	IR camera and Depth camera calibration provided

Amin, D., Govilkar, S. Comparative Study of Augmented Reality SDK's, Int'l Journal on Computational Sciences & Applications, 5(1): 11-26, February 2015.

Based on Overlaying Capability

AR SDK Type		Vuforia	Metaio	Wikitude	ARToolKit	D'Fusion	ARmedia
Overlaying capability	2D content	✓	✓	✓	✓	✓	✓
	3D content	✓	✓	✓	✓	✓	✓
	Others	3D animation can be overlaid on screen	Billboard can be overlaid	Sprite animations, 3D transformations and HTML contents	Support high level graphic content and animations.	3D animation can be overlaid on screen	Interactive 3D animation through SketchUp

Amin, D., Govilkar, S. Comparative Study of Augmented Reality SDK's, Int'l Journal on Computational Sciences & Applications, 5(1): 11-26, February 2015.

Benefits of Different AR SDK's

Benefits	
Vuforia	Enable to maintain tracking even when the target is out of view and view them from greater distance. Cloud Database allows storing thousands of image targets.
Metaio	Powerful 3D rendering engine with capability load 3D model of .obj format. No limit on number of trackable object depends on device memory.
Wikitude	AR content can be programmed using basic HTML5, JavaScript and CSS. Easy portability of AR apps from one platform to another.
ARToolKit	Multiple platforms AR app development possible. Only AR SDK which is available as open source, through which many new AR frameworks developed.
D'Fusion	High quality 3D content can be augmented with support for multiple 3D object formats. Provides encrypted media to prevent privacy or substitution risks.
ARmedia	Depth camera calibration provided which creates more immersive experience

Amin, D., Govilkar, S. Comparative Study of Augmented Reality SDK's, Int'l Journal on Computational Sciences & Applications, 5(1): 11-26, February 2015.

Limitation of Different AR SDK's

	Limitation
Vuforia	Vuforia SDK for Android does not expose any utility function to easily load a 3D model from any standard format. Device database can only support 100 image targets.
Metaio	Difficult to render complex 3D objects also limitation is associated with model size.
Wikitude	Doesn't track 3D model which limits is use to only 2D tracking. Target image to track need to be of solid colors to be recognized
ARToolKit	Less accuracy in tracking markers even when camera and marker are still. It itself doesn't support location based augmented reality
D'Fusion	Video file supported but audio associated with video can't be played
ARmedia	Doesn't support all type of textures for 3D objects

Amin, D., Govilkar, S. Comparative Study of Augmented Reality SDK's, Int'l Journal on Computational Sciences & Applications, 5(1): 11-26, February 2015.

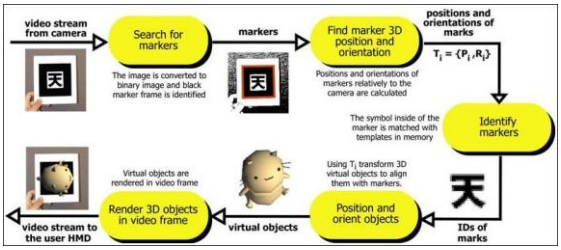
ARToolKit Basics

How Does ARToolKit Work

- The camera captures video of the real world and sends it to the computer
- Software on the computer searches through each video frame for any square shapes
- If a square is found, the software uses some mathematics to calculate the position of the camera relative to the black square
- Once the position of the camera is known a computer graphics model is drawn from that same position
- This model is drawn on top of the video of the real world and so appears stuck on the square marker
- The final output is shown back in the handheld display, so when the user looks through the display they see graphics overlaid on the real world

http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm

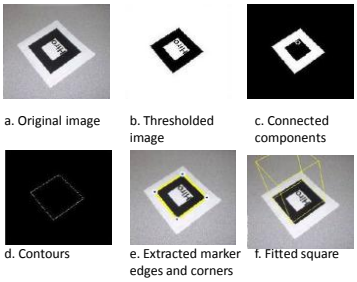
How Does ARToolKit Work .



http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

Computer Vision Algorithm

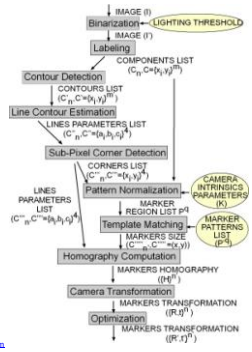
- Based on a basic corner detection approach with a fast pose estimation algorithm



a. Original image
b. Thresholded image
c. Connected components
d. Contours
e. Extracted marker edges and corners
f. Fitted square

http://www.hitl.washington.edu/artoolkit/documentation/vision.htm

Computer Vision Algorithm



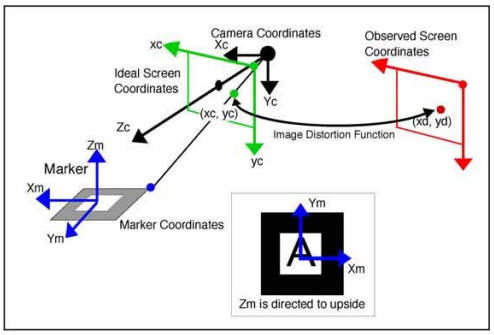
http://www.hitl.washington.edu/artoolkit/documentation/vision.htm

ARToolKit Coordinate Systems

- ARToolKit defines different coordinate systems mainly used by the
  - Computer vision algorithm
  - Rendering
- Need to understand relations between them
  - To prevent reverse image or bad position object display

http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

Computer Vision Coordinate System



http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

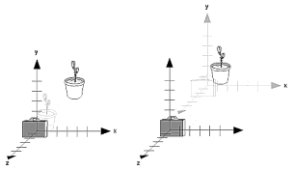
Computer Vision Coordinate System

- arGetTransMat give the position of the marker in the camera coordinate system
  - Not the reverse
- Tip:
  - If you want the position of the camera in the marker coordinate system you need to inverse this transformation
    - arMatrixInverse()

http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

Rendering Coordinate System

- When you use ARToolKit with OpenGL, remember that OpenGL is a right-handed coordinate system, with Z coming to you
  - i.e. the camera is facing in the direction of -Z



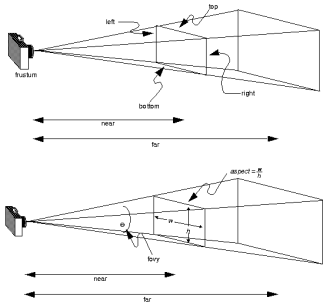
http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

Rendering Coordinate System .

- ARToolKit uses a calibrated camera perspective that results in an off-axis projection matrix for OpenGL
  - Can not be created by calling gluPerspective, but requires the extra parameters of glFrustum
- Rather than decomposing ARToolKit's projection into parameters to pass to glFrustum, directly load the OpenGL projection matrix by:
  - Setting glMatrixMode(GL_PROJECTION_MATRIX)
  - Calling glLoadMatrix

http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

Rendering Coordinate System ..



http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

ARToolKit Framework

Introduction to Framework

- ARToolKit is a software Toolkit like GLUT
- It furnishes predefined functions that need to call in a specific order for developing an AR application
  - Can also use different parts of the Toolkit separately
- ARToolKit supports multiple platforms
  - While attempting to minimise library dependencies without sacrificing efficiency

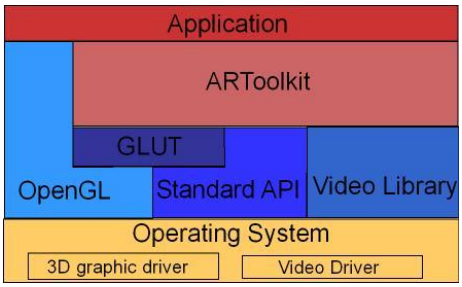
http://www.hitl.washington.edu/artoolkit/documentation/devframework.htm

Introduction to Framework .

- ARToolKit uses
  - OpenGL for the rendering part
  - GLUT for the windows/event handler aspect
  - Hardware-dependent video library
  - Standard API on each platform
- The API is in C and a lot of samples provide good starting points for creating new applications
  - Skeleton framework

http://www.hitl.washington.edu/artoolkit/documentation/devframework.htm

ARToolKit Architecture



http://www.hitl.washington.edu/artoolkit/documentation/devframework.htm

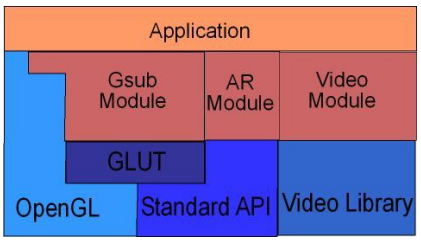
ARToolKit Structure

- The ARToolKit library consists of four modules:
  - AR module: core module with marker tracking routines, calibration and parameter collection
  - Video module: a collection of video routines for capturing the video input frames
  - Gsub module: a collection of graphic routines based on the OpenGL and GLUT libraries
  - Gsub_Lite module: replaces GSub with a more efficient collection of graphics routines, independent of any particular windowing toolkit

http://www.hitl.washington.edu/artoolkit/documentation/devframework.htm

Hierarchical Structure of ARToolKit

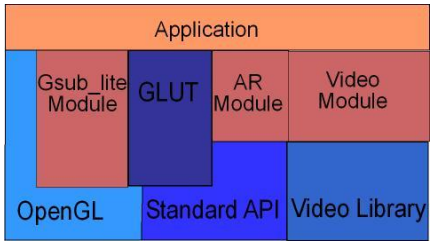
- Hierarchical structure of ARToolKit using Gsub Module



http://www.hitl.washington.edu/artoolkit/documentation/devframework.htm

Hierarchical Structure of ARToolKit .

- Hierarchical structure of ARToolKit using Gsub_Lite Module



http://www.hitl.washington.edu/artoolkit/documentation/devframework.htm

ARToolKit Pipeline

- The modules respect a global pipeline metaphor
  - i.e. video->tracking->display
- User can easily replace any module with another
  - i.e. gsub with Open Inventor renderer

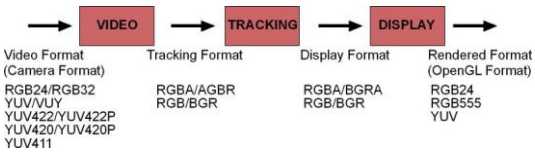
http://www.hitl.washington.edu/artoolkit/documentation/devframework.htm

Data Types

- ARToolKit manipulates a lot of different kinds of variables
- Internally, it uses global variables, that restrict re-entrant part of the code
- Otherwise, standard multi-argument interface are used based on a data-flow approach
- ARToolKit uses different image formats between different modules

http://www.hitl.washington.edu/artoolkit/documentation/devframework.htm

ARToolKit Data-Flow



- Information about the detected markers is contained in the ARMarkerInfo structure defined in ar.h in the include directory

http://www.hitl.washington.edu/artoolkit/documentation/devframework.htm

ARToolKit Documentation

- All ARToolKit functions generally begin with the prefix ar, and a specific prefix for each module
  - arVideo for video
  - arg for graphics
  - ar for main core
- More information:
  - http://artoolkit.sourceforge.net/apidoc/index.html

http://www.hitl.washington.edu/artoolkit/documentation/devframework.htm

ARToolKit Hardware

ARTooKit Camera

- The choice of a camera is the most important
  - Resolution, frame-rate, optics distortion, are some of the main parameters for selected your camera
- But supported platforms, parameters of the driver (like auto-contrast) can also influence your choice
- Some cameras enable by default auto-contrast that decrease your performance, or offers 25Hz frame-rate with a lost of image quality
  - i.e. distortion, latency lag

http://www.hitl.washington.edu/artoolkit/documentation/hardware.htm

ARTooKit Camera

- The most efficient solution keep a video capture card with a PAL or NTSC camera that offers a large choice and a really good quality camera
  - But a PAL camera can be really expensive according its characteristics
- The signal deliver is RGB image (color palette) that reduce hardware or software palette conversion cost

http://www.hitl.washington.edu/artoolkit/documentation/hardware.htm

ARTooKit Camera

- The traditional choice is USB or Firewire camera
- Can notice that frame-rate or color palette (RGB, YUV, YUV compressed) depend mainly on the bandwidth of the technology
- USB Camera used generally compressed format transmission like YUV:4:2:2 YUV:4:1:1
  - i.e. lossy compression
- Firewire Cameras offers better solution, but camera with full RGB color palette are generally expensive
  - A compressed format in VGA remains a good choice

http://www.hitl.washington.edu/artoolkit/documentation/hardware.htm

ARTooKit Camera Table 1

USB1.1	max 1.5 MByte/s	most low cost solution.
IEEE1394a	60 Mbyte/s	good solution with a standardized camera protocol.
PCI32Bit1.0 (33Mhz)	125.89 MByte/s	
PCI64Bit2.0 (33Mhz)	251.77 MByte/s	
USB2.0	max 50 MByte/s	badly supported on Linux.
IEEE1394b	100 Mbyte/s	few cameras support this protocol.
PCI32Bit2.1 (66Mhz)	251.77 MByte/s	
PCI64Bit2.1 (66Mhz)	503.54 MByte/s	

http://www.hitl.washington.edu/artoolkit/documentation/hardware.htm

ARTooKit Camera Table 2

SIF RGB 15 fps	27 MBit/s (3.37 MByte/s)
SIF RGB 30 fps	55 MBit/s (6.87 MByte/s)
SIF YUV 4:1:1 15 fps	13 MBit/s (1.62 MByte/s)
SIF YUV 4:1:1 30 fps	27 MBit/s (3.37 MByte/s)
VGA RGB 15 fps	106 MBit/s (13.25 MByte/s)
VGA RGB 30 fps	221 MBit/s (26.37 MByte/s)
VGA YUV 4:1:1 15 fps	53 MBit/s (6.63 MByte/s)
VGA YUV 4:1:1 30 fps	106 MBit/s (13.25 MByte/s)

http://www.hitl.washington.edu/artoolkit/documentation/hardware.htm

HMDs and ARTooKit

- ARToolKit uses computer vision techniques for image-based recognition and tracking
- Since it uses only a single camera, a self-contained head tracking system can be developed if this camera is fixed to an HMD
- In this way AR applications can be developed which use HMDs

http://www.hitl.washington.edu/artoolkit/documentation/hardware.htm

Non-HMD and ARTooKit

- It is not necessary to have a head mounted display to use the ARToolKit
  - A camera connected to a computer is the only requirement
- Without an HMD ARToolKit applications can be viewed on a computer monitor
  - With an HMD a more immersive experience can be created

http://www.hitl.washington.edu/artoolkit/documentation/hardware.htm

ARToolKit Limitations and Benchmarks

General Limitations

- There are some limitations to purely computer vision based AR systems
- Naturally the virtual objects will only appear when the tracking marks are in view
  - This may limit the size or movement of the virtual objects
- It also means that if users cover up part of the pattern with their hands or other objects the virtual object will disappear

http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

Limitations Range

- The larger the physical pattern the further away the pattern can be detected and so the great volume the user can be tracked in

Pattern Size (inches)	**Usable Range** (inches)
2.75	16
3.50	25
4.25	34
7.37	50

http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

Limitations Pattern Complexity

- The simpler the pattern the better. Patterns with large black and white regions are the most effective
  - i.e. low frequency patterns
- Replacing a 4.25 inch square pattern with a pattern of the same size but much more complexity will reduce the tracking range from 34 to 15 inches

http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

Limitations Marker Orientation

- Tracking is also affected by the marker orientation relative to the camera
- As the markers become more tilted and horizontal, less and less of the center patterns are visible and so the recognition becomes more unreliable

http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

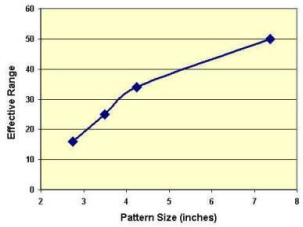
Limitations Lighting

- Overhead lights may create reflections and glare spots on a paper marker and so make it more difficult to find the marker square
- To reduce the glare patterns can be made from more non-reflective material
  - For example, by gluing black velvet fabric to a white base
- The 'fuzzy' velvet paper available at craft shops also works very well

http://www.hitl.washington.edu/artoolkit/documentation/cs.htm

Pattern Size Parameter

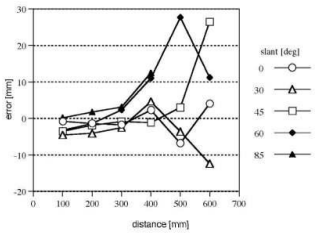
- Tracking Error with Pattern Size



http://www.hitl.washington.edu/artoolkit/documentation/benchmark.htm

Range Parameter

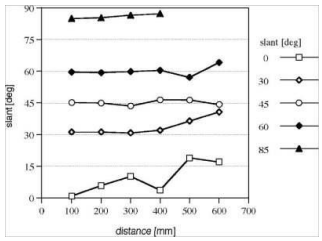
- Tracking Error with Range



http://www.hitl.washington.edu/artoolkit/documentation/benchmark.htm

Angle Parameter

- Tracking Error with Angle



http://www.hitl.washington.edu/artoolkit/documentation/benchmark.htm

First ARToolKit Example

First Step

- Once ARToolKit has been installed there is a sample program, simpleTest or simple according your ARToolKit version, in the bin directory that can be run to show the capabilities of ARToolKit
- To run the code you need to print out the hiroPatt.pdf paper fiducial marker that is contained in the directory patterns
  - Best performance is achieved if this is glued to a piece of cardboard to keep it flat

http://www.hitl.washington.edu/artoolkit/documentation/userstartup.htm

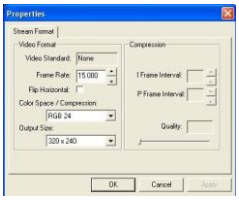
Running ARToolKit

- In each platform you have generally two choices:
  - Click on the program from your OS explorer
  - Starting from the command line
    - The last choice is better since it give you the error and standard output stream (and ARToolKit used it a lot)
- Each platform offer a dialog box to setup the video before starting the main AR loop

http://www.hitl.washington.edu/artoolkit/documentation/userstartup.htm

Windows

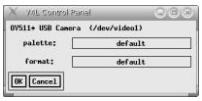
- On a Windows PC double click on the simple.exe icon in the bin directory from your windows explorer
- A dos console window will open and when the camera is detected the follow dialog will open



http://www.hitl.washington.edu/artoolkit/documentation/userstartup.htm

Linux

- On a Linux double click on the simple icon in the bin directory from your gnome or kde explorer
  - Notice than you will not have error or output stream display
- Otherwise start a terminal, go the bin directory and runsimple
  - If you have V4L this dialog will be appear



http://www.hitl.washington.edu/artoolkit/documentation/userstartup.htm

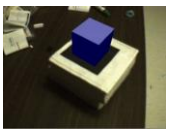
MacOS

- On MacOS X double click on the simple icon in the bin directory from your mac explorer
- A console window will open and when the camera is detected the follow dialog will open
- Otherwise start the Terminal program, go the bin directory and run simple

http://www.hitl.washington.edu/artoolkit/documentation/userstartup.htm

simpleTest Output

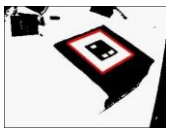
- In order for the virtual object to appear the entire black square border of the marker must be in view and also the pattern contained within the square border must be visible at all times
- If the virtual image does not appear, or it flickers in and out of view it may be because of lighting conditions
- This can often be fixed by changing the lighting threshold value used by the image processing routines



http://www.hitl.washington.edu/artoolkit/documentation/userstartup.htm

simpleTest Output

- If you hit the ‘t' key on the keyboard you will be prompted to enter a new threshold value
- This should be a value between 0 and 255; the default is 100
- Hitting the ‘d' key will show the thresholded video image below the main video window
- Possible tracking patterns found in the input image are marked by a red square in the thresholded image
- This will help you check the effect of lighting and threshold values on the video input



http://www.hitl.washington.edu/artoolkit/documentation/userstartup.htm

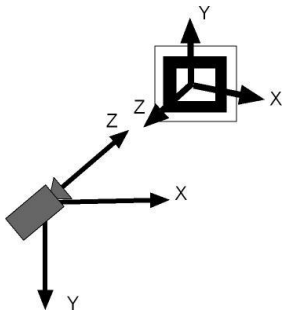
ARToolKit Tutorials

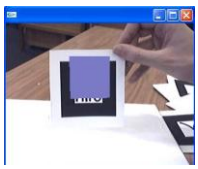
Camera and Marker Relationships

- ARToolKit gives the position of the marker in the camera coordinate system, and uses OpenGL matrix system for the position of the virtual object
- These different elements and their relationship is explained in the next slides

http://www.hitl.washington.edu/artoolkit/documentation/tutorialcamera.htm

ARToolKit's Coordinate System

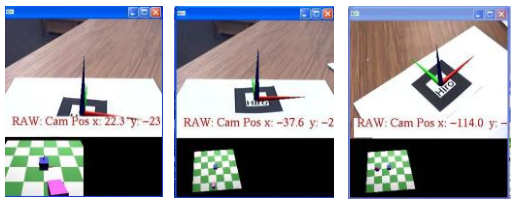




http://www.hitl.washington.edu/artoolkit/documentation/tutorialcamera.htm

'exview' Example

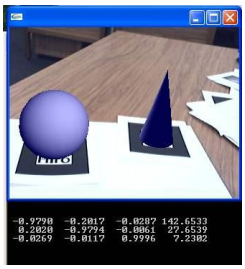
- The example demonstrates:
  - The camera position in the marker (in 3D and text)
  - A 3D model of the camera and the marker



http://www.hitl.washington.edu/artoolkit/documentation/tutorialcamera.htm

'relationTest' Example

- Computes the relative transformation between two markers
  - Important for having different scenarios



http://www.hitl.washington.edu/artoolkit/documentation/tutorialcamera.htm

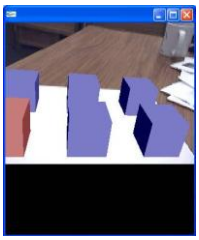
Multi-Marker Tracking

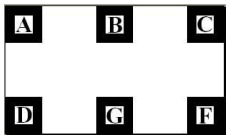
- ARToolKit can give you the position of multiple markers as a function of the camera coordinate system
- Can also have a set of markers that define only one position
  - i.e. Glue to a cardboard plane
- ARToolKit can do that with a specific set of functions based on the multiMarker module

http://www.hitl.washington.edu/artoolkit/documentation/tutorialmulti.htm

Using Multiple Markers

- Print the pattMulti.pdf





- Run multiTest example

http://www.hitl.washington.edu/artoolkit/documentation/tutorialmulti.htm

'multiMarker' Configuration File

```
#the number of patterns to be recognized
6

#marker 1
Data/multi/patt.a
40.0
0.0 0.0
1.0000 0.0000 0.0000 -100.0000
0.0000 1.0000 0.0000 50.0000
0.0000 0.0000 1.0000 0.0000
...
```

Pattern File

Pattern Width + Coordinate Origin

Pattern Transform Relative to Global Origin

http://www.hitl.washington.edu/artoolkit/documentation/tutorialmulti.htm

ARToolKit Code

Video Initialization

- Configure the video input
  - vconf = <video configuration string>
- Start video capture
  - arVideoCapStart();
- In init(), open the video
  - arVideoOpen(vconf);
  - arVideoInqSize(&xsize, &ysize);
- When finished, close the video path
  - arVideoCapStop();
  - arVideoClose();

Changing Image Size

- For input capture
  - vconf = "videoWidth=320,videoHeight=240";
    - Note – the camera must support this image size
- For display
  - argInit(&cparam, 1.5, 0, 0, 0, 0);
    - The second parameter means zoom ratio for display image size related to input image

Graphics Handling: libARgsub

- Set up and clean up the graphics window

```
void argInit( ARParam *cparam, double zoom,
        int fullFlag, int xwin, int ywin, int hmd_flag );
void argCleanup( void );
```

  - cparam: camera parameter
  - zoom: zoom ratio
  - fullFlag: 0: normal, 1: full screen mode
  - Xwin, ywin: create small window for debug
  - hmd_flag: 0: normal, 1: optical see-through mode

Main Function

```
main()
{
   init();
   argMainLoop( mouseEvent,
   keyEvent, mainLoop);
}
```

Graphics handling: libARgsub

- Go into the iterative cycle

```
void argMainLoop(
   void (*mouseFunc)(int btn,int state,int x,int y),
   void (*keyFunc)(unsigned char key, int x, int y),
   void (*mainFunc)(void)
);
```

- Swap buffers
  - void argSwapBuffers(void);

mainLoop Function

```
if( dataPtr = (ARUint8 *)
arVideoGetImage()) == NULL ) {
arUtilSleep(2);
return;
}
argDrawMode2D();
argDispImage(dataPtr, 0, 0 );
arVideoCapNext();
argSwapBuffers();
```

Image capture: libARvideo

- Return the pointer for captured image
  - ARUint8 *arVideoGetImage(void);
- Pixel format and byte size are defined in config.h
  - #define AR_PIX_FORMAT_BGR
  - #define AR_PIX_SIZE 3

Graphics handling: libARgsub

- Set the window for 2D drawing
  - void argDrawMode2D(void);
- Set the window for 3D drawing
  - void argDrawMode3D(void);
  - void argDraw3dCamera(int xwin, int ywin);
- Display image
  - void argDispImage(ARUint8 *image, int xwin, int ywin);

Detecting a Marker

- Key points:
  - Threshold value
  - Important external variables
  - arDebug – keep thresholded image
  - arImage – pointer for thresholded image
  - arImageProcMode – use 50% image for image processing
    - AR_IMAGE_PROC_IN_FULL
    - AR_IMAGE_PROC_IN_HALF

Marker Detection

```
/* detect the markers in the video frame */
if(arDetectMarker(dataPtr, thresh,
&marker_info, &marker_num) < 0 ) {
    cleanup();
    exit(0);
}

for( i = 0; i < marker_num; i++ ) {
   argDrawSquare(marker_info[i].vertex,0,0);
}
```

Marker Structure

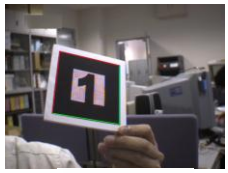
```
typedef struct {
    int area;
    int id;
    int dir;
    double cf;
    double pos[2];
    double line[4][3];
    double vertex[4][2];
} ARMarkerInfo;
```

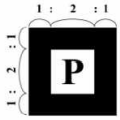
Using a Pattern

- Key points
  - Pattern files loading
  - Structure of marker information
    - Region features
    - Pattern Id, direction
    - Certainty factor
  - Marker identification

Making a Pattern Template

- Use of utility program:
  - mk_patt.exe
- Show the pattern
- Put the corner of red line segments on the left-top vertex of the marker
- Pattern stored as a template in a file
- 1:2:1 ratio determines the pattern region used





Pattern File Loading

```
int patt_id;
char *patt_name = “Data/kanjiPatt”

/* load pattern file */
if(patt_id=arLoadPatt (patt_name) < 0)
{
printf ("Pattern file load error !! \n");
exit(0);
}
```

Checking for Known Patterns

```
/* check for known patterns */
k = -1;

for( i = 0; i < marker_num; i++ ) {
    if( marker_info[i].id == patt_id) {
    /* you've found a pattern */
    printf("Found pattern: %d \n",patt_id);
          if( k == -1 ) k = i;
    else
          /* make sure you have the best pattern (highest confidence
          factor) */
    if( marker_info[k].cf < marker_info[i].cf )
    k = i;
    }
}
```

Getting 3D Information

- Key points
  - Definition of a real marker
  - Transformation matrix
    - Rotation component
    - Translation component

Transformation Matrix

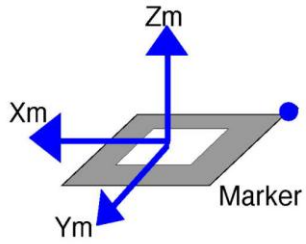
```
double marker_center[2] = {0.0, 0.0};
double marker_width = 80.0;
double marker_trans[3][4];

arGetTransMat(&marker_info[i],
  marker_center, marker_width,
  marker_trans);
```

Finding the Camera Position

- This function sets transformation matrix from marker to camera into marker_trans[3][4]
  - arGetTransMat(&marker_info[k], marker_center, marker_width, marker_trans);
- You can see the position information in the values of marker_trans[3][4]
  - Xpos = marker_trans[0][3];
  - Ypos = marker_trans[1][3];
  - Zpos = marker_trans[2][3];

ARToolKit Coordinate Frame



Virtual Object Display

- Key points
  - OpenGL parameter setting
  - Setup of projection matrix
  - Setup of modelview matrix

Appending your Own OpenGL Code

- Set the camera parameters to OpenGL Projection matrix
  - argDrawMode3D();
  - argDraw3dCamera(0, 0);
- Set the transformation matrix from the marker to the camera to the OpenGL ModelView matrix
  - argConvGlpara(marker_trans, gl_para);
  - glMatrixMode(GL_MODELVIEW);
  - glLoadMatrixd(gl_para);
- After calling these functions, your OpenGL objects are drawn in the real marker coordinates

3D Model Loading

- ARToolKit does not have a function to handle 3D CG models
- 3rd party rendering software should be employed
  - OpenVRML
  - OpenSceneGraph
  - 3D parsers
  - Unity
  - etc

Loading Multiple Patterns

- Object Structure

```
typedef struct {
        char name[256];
        int id;
        int visible;
        double marker_coord[4][2];
        double trans[3][4];
        double marker_width;
        double marker_center[2];
} ObjectData_T;
```

Finding Multiple Transforms

- Create object list
  - ObjectData_T *object;
- Read in objects - in init( )
  - read_ObjData(char *name, int *objectnum);
- Find Transform – in mainLoop( )

```
for( i = 0; i < objectnum; i++ ) {
        // Check patterns
        // Find transforms for each marker
}
```

Drawing Multiple Objects

- Send the object list to the draw function
  - draw(object, objectnum);
- Draw each object individually

```
for( i = 0; i < objectnum; i++ ) {
        if( object[i].visible == 0 ) continue;
        argConvGlpara(object[i].trans, gl_para);
        draw_object( object[i].id, gl_para);
}
```

ARToolKit Examples

iNVENtor 2014 AR Showcase

https://www.youtube.com/watch?v=pVyFxdJSIcI

ARToolKit + iPhone Video

https://www.youtube.com/watch?v=5M-qAmBDc2k

ARToolKit + Solar System

https://www.youtube.com/watch?v=OKcWp7NLVz4

Conclusions

- Although a lot of solutions exist, there is no complete solution for all types of AR
- Depending on the app, specific SDK/solution should be used
- Expect in the future more 'complete' tools

Questions

Acknowledgements

- Special Thanks to Prof. Mark Billinghurst