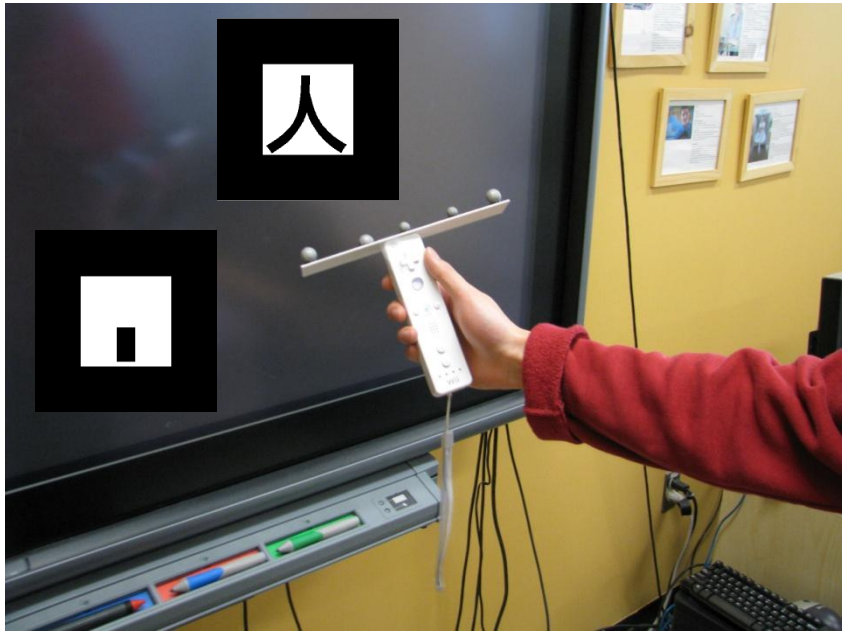
Vision Recognition for Mixed Reality Applications

Richard Fung

October 2010

With annotations

Vision Recognition System



3. Application

2. Vision Recognition

1. Webcam Capture

Webcam

Some work better than others…

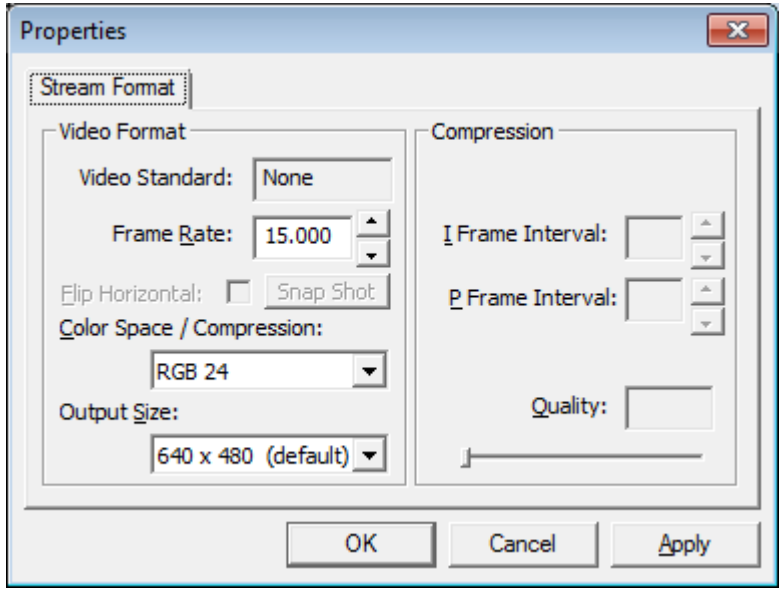
Logitech C905
QuickCam Pro for Notebooks

- Tried and tested
  - Design Interfaces Lab (Dr. Igarashi)
  - MIT Media Lab
- Characteristics
  - Wide angle lens
  - Auto focus
  - 800 x 600 @ 30 fps
  - 1600 x 1200 @ 10 fps

Webcam

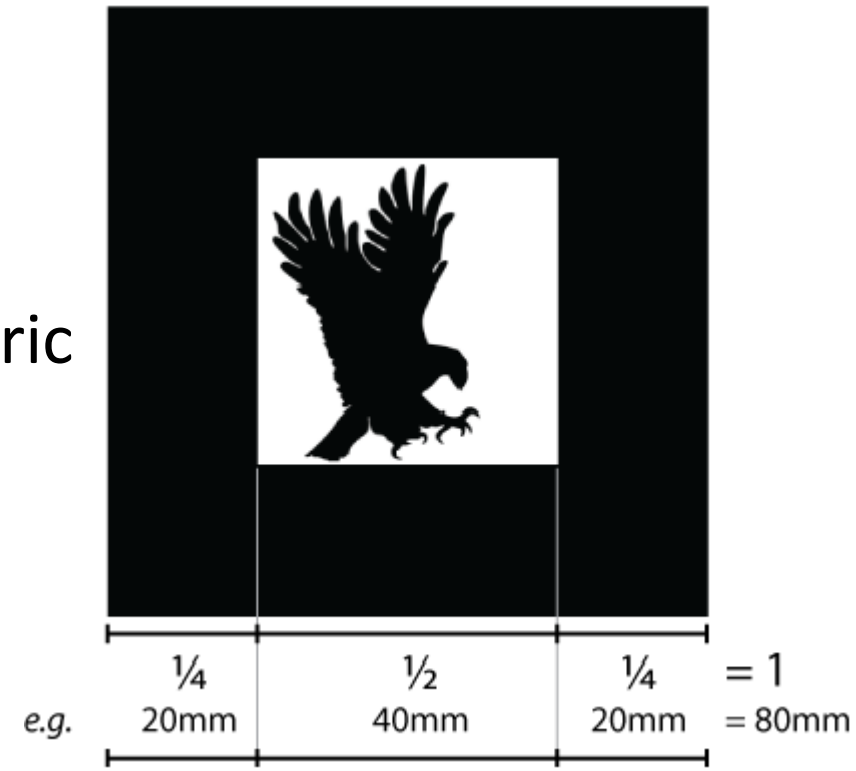
VirtualDub:

File > Capture AVI
Device > DirectShow
Video > Capture Pin



Markers

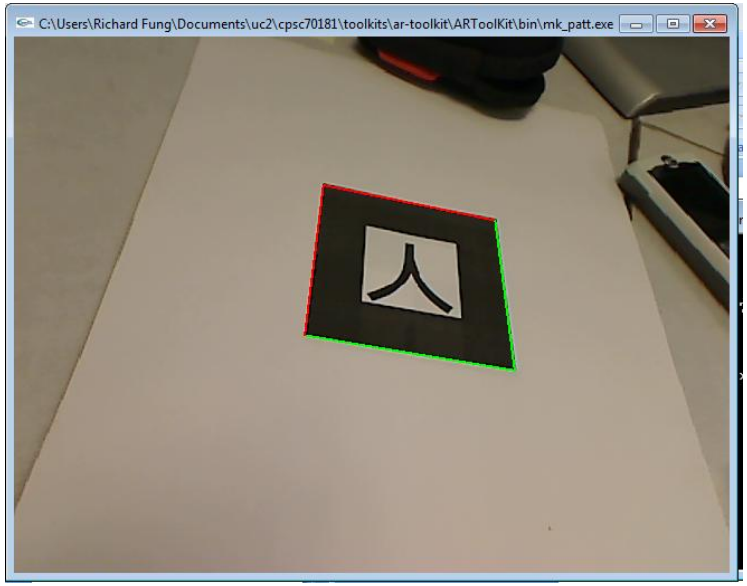
- Characteristics
  - Square
  - Continuous border
  - Not rotationally symmetric
  - Black border:



ARToolkit and variants

Markers

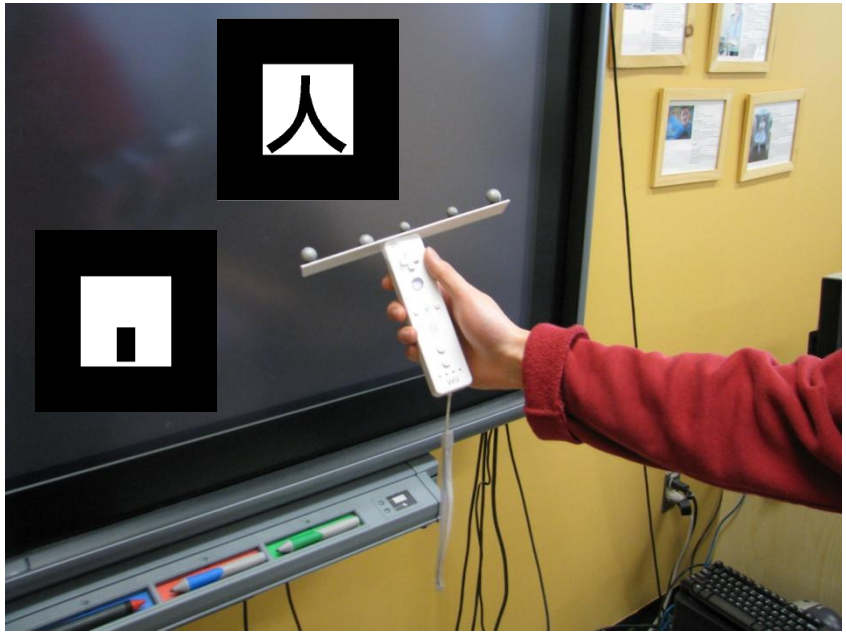
- Pattern design
  - From webcam
    - ARToolkit/bin/mk_patt.exe
  - Use existing patterns
    - ARToolkit/patterns
      - Hiro
      - Kanji
    - Search the web

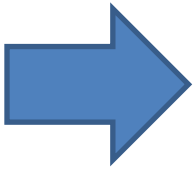


Markers

- Issues
  - Size of marker
    - further away = larger
  - Complexity of pattern
    - False positives
  - Reflection of light
    - Use felt instead of laser print
    - Print at low toner option

Vision Recognition System





3. Application

2. Vision Recognition

1. Webcam Capture

Various Packages

• ARToolkit	C++	open/proprietary
• ARToolkitPlus	C++	open
• ARTag	C++	proprietary
• NyARToolkit	Java, C#	open
• Goblin XNA	C#	open
• Igarashi	Java, C#	proprietary

NyARToolkit

- Setup
  - Calibration
  - Create marker definition files
- Issues
  - Incomplete
  - Documentation in Japanese

NyARToolkit

- Source code
  - Distribution 2.5.2.1
  - Added a new method: NyARDetectMarker.getCorners()
  - Download from iLab Cookbook

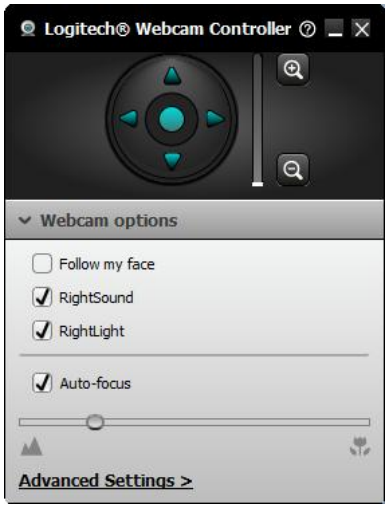
Dependencies

- Visual Studio 2008 or later
- Microsoft DirectX SDK 2002 to August 2007
  - Requires Managed Direct X 1.1
- Webcam
  - Next slide…

Webcam Capture

- Windows: Microsoft DirectShow
  - API provided by toolkit

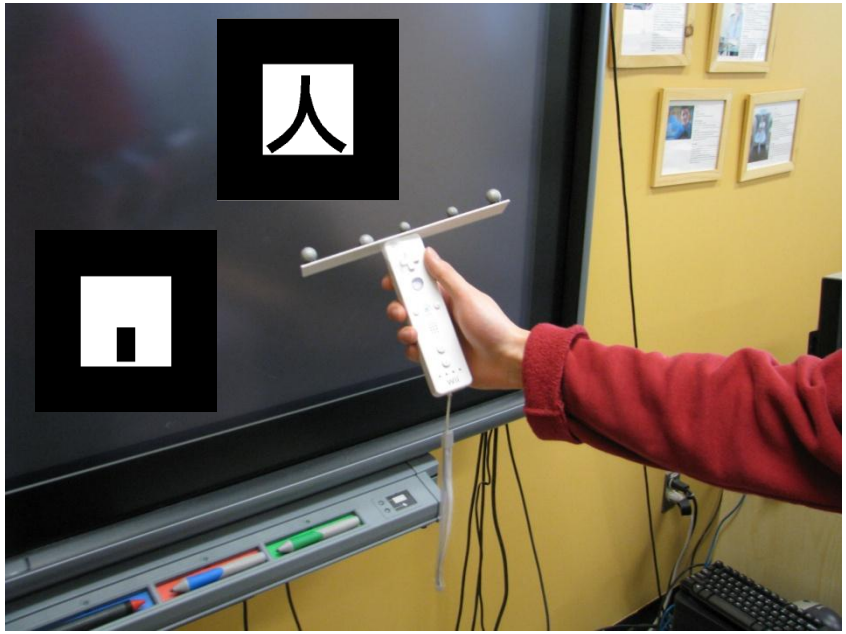
- Issues
  - Auto focus blur
  - Image: Contrast, lighting, flickering
  - Colour image vs. B&W




Logitech® Webcam Properties
Logitech Webcam 905
Webcam Control
Advanced Settings
Device Info
Follow my face
RightSound
Find my face (for widescreen viewing)
Auto-focus
Logitech
Restore to defaults
Save
Cancel


Logitech® Webcam Properties
Logitech Webcam 905
Webcam Control
Advanced Settings
Device Info
Image Quality:
RightLight
Exposure
Gain
Auto
Brightness
Contrast
Color Intensity
Anti-flicker:
Off
NTSC - 60Hz
PAL- 50Hz
Image orientation:
Normal
Mirrored
Logitech
Restore to defaults
Save
Cancel

Vision Recognition System



3. Application

2. Vision Recognition

1. Webcam Capture

Vision Recognition

- Runtime
  - Initialization
    - Select a webcam ← next slide…
    - Load webcam parameters to NyARToolkit
    - Load pattern(s) to detect

```
using NyARToolkitCSUtils.Capture;
using NyARToolkitCSUtils.NyAR;

 public partial class MainWindow : Window, CaptureListener
{
        private CaptureDevice m_cap;

        public MainWindow()
        {
                […]
                CaptureDeviceList cl = new CaptureDeviceList();
                CaptureDevice cap = cl[0];
                m_cap.SetCaptureListener(this);
                m_cap.PrepareCapture(800, 600, 30);
                […]
        }

        void CaptureListener.OnBuffer(CaptureDevice i_sender, double i_sample_time,
                                      IntPtr i_buffer, int i_buffer_len)
        {
        }
```

Callback interface

Enumerate webcams

Width, height, frame rate

Callback method

```
using jp.nyatla.nyartoolkit.cs;
using jp.nyatla.nyartoolkit.cs.core;
using jp.nyatla.nyartoolkit.cs.detector;

public partial class MainWindow : Window, CaptureListener
{
        private NyARDetectMarker m_ar;

        public MainWindow()
        {
                […]
                //AR用カメラパラメタファイルをロード
                // AR camera parameters file to read
                NyARParam ap = new NyARParam();
                ap.loadARParamFromFile("../../data/camera_para.dat");
                ap.changeScreenSize(800, 600);

                //AR用のパターンコードを読み出し
                //AR's pattern to detect from the webcam
                NyARCode code = new NyARCode(16, 16);
                code.loadARPattFromFile("../../data/patt.kanji");
                this.m_ar = new NyARDetectMarker(ap,
                        new NyARCode[]{code}, new double[]{80.0}, 1,
                        NyARBufferType.BYTE1D_B8G8R8_24);
                this.m_ar.setContinueMode(false);
```

Detection object

Webcam Calibration

Width, height again

16 blocks inside black border

80 mm marker size

Recognize on a 24 bpp bitmap

Vision Recognition

- Runtime
  - New frame event
    - B & W threshold
    - Marker detection ← next slide…
    - Application logic
  - Better design
    - Marker detection + logic in a timer loop

```
void CaptureListener.OnBuffer(CaptureDevice i_sender, double i_sample_time,
                              IntPtr i_buffer, int i_buffer_len)
        {
            // calculate size of the frame bitmap
            int w = i_sender.video_width;
            int h = i_sender.video_height;
            int s = w * (i_sender.video_bit_count / 8); // stride

            // thresholding […]

            // detectMarkerLite requires BGR image from DirectX
            // m_threshold 0-255
            int detectedMkrs = this.m_ar.detectMarkerLite(ra, m_threshold);

            // save the result of the detection
            NyARSquare square = null;
            if (detectedMkrs > 0)
            {
                // vertices of the square are returned
                NyARTransMatResult transMat = new NyARTransMatResult();
                NyARDoublePoint2d[] points = m_ar.getCorners(0); // RichF
                square = new NyARSquare();
                square.sqvertex = points;
            }
```

square ← application logic for recognized marker

```
void CaptureListener.OnBuffer(CaptureDevice i_sender, double i_sample_time,
                              IntPtr i_buffer, int i_buffer_len) {
[…]
AForge.Imaging.Filters.FiltersSequence seq =
                                        new AForge.Imaging.Filters.FiltersSequence();

// order here is important, pipe and filters design pattern
seq.Add(new AForge.Imaging.Filters.Grayscale(0.2125, 0.7154, 0.0721));
seq.Add(new AForge.Imaging.Filters.Threshold(127));
seq.Add(new AForge.Imaging.Filters.GrayscaleToRGB()); // 24 bit image

// run the threshold algorithm
AForge.Imaging.UnmanagedImage srcImg =
         new AForge.Imaging.UnmanagedImage(i_buffer, w, h, s,
         System.Drawing.Imaging.PixelFormat.Format32bppRgb);
AForge.Imaging.UnmanagedImage outputImg = seq.Apply(srcImg);
```

Grayscale
+ B/W threshold

Original camera
Image from
DirectShow is 32 bpp

Move Aforge.imaging bitmap → NyARToolkit bitmap

```
[…] thresholding

// load a RGB buffer into the NyAR format, which we copy from AForge.Imaging
NyARRgbRaster_RGB ra = new NyARRgbRaster_RGB(w, h, false);
byte[] destArr = new byte[outputImg.Stride * outputImg.Height];
System.Runtime.InteropServices.Marshal.Copy(outputImg.ImageData, destArr,
                                                0, outputImg.Stride * outputImg.Height);
ra.wrapBuffer(destArr);

// detectMarkerLite requires BGR image from DirectX
// m_threshold 0-255
int detectedMkrs = this.m_ar.detectMarkerLite(ra, m_threshold);

[…] application logic
```

Examples

- WPF:
  - My example

- Packaged examples:
  - NyARToolkitCS-2.5.2.1\forFW2.0\sample

Examples

- RawTest
  - bitmap data + recognizer
- CaptureTest
  - webcam + recognizer + System.Drawing.Bitmap
- SimpleLiteDirect3d → 2 patterns
- SingleNyIdMarkerDirect3d → 1 pattern
  - webcam + recognizer + Direct3D

Where to go?

- ARToolkitPlus — Fudicials with built-in ID
- Goblin XNA — Extends ARTag
.NET
- Igarashi Detection — Orthogonal viewing
Smaller marker sizes

Thanks

- Miaosen Wang — ARToolkit example
- Paul Lapides — webcam info & ARToolkitPlus