ARTag Rev2 Fiducial Marker System: Vision-based Tracking for AR

Mark Fiala
Computational Video Group
Institute of Information Technology
National Research Council
mark.fiala@nrc-cnrc.gc.ca

www.artag.net
Talk Overview

• Fiducial Marker Systems
• Fiducial Marker Performance Criteria/Metrics
• What is not a fiducial marker
• ARToolkit, ARTag, ARToolkit Plus
• Comparing ARTag and ARToolkit, ARToolkit Plus
• New SDK: Use ARTAG: Rev2 functions
• Use ARTAG: Rev1 functions
• ARTag Drop-In Functions
• Demos
Computer Vision Image Features

- markerless (interest points: SIFT, PCA-SIFT, …)
- marker-based = **Fiducial Marker Systems**

**Fiducial Marker Systems**

**Consist of:**

1. Library of patterned markers to mount in environment (on target)
2. Computer vision algorithm to find projection of markers in digital camera image
Fiducial Marker Performance Criteria/Metrics

- false positive rate: how often is a marker erroneously reported
- false negative rate: how often is a marker missed
- inter-marker confusion rate: how often is one marker mistaken for another
- lighting immunity: performance under harsh uncontrolled lighting
- occlusion immunity: does marker have to be completely visible for detection
- perspective/affine projection support
- planarity restriction: markers on flat or curved/warped surfaces
- library size: how many unique markers can be handled
- minimum & maximum image size -> range of distances for detection
- photometric calibration required
- speed performance: processing requirements
What is not a fiducial marker

• Why not use standard barcodes?

• Not suitable as a **fiducial marker** because:
  • Must be parallel to camera, in known rotation
  • Only provides position in one direction
  • Resolution requirement too high. >500,1000 linear pixels
What is not a fiducial marker: 2D planar patterns

• Not suitable as a *fiducial marker* because:
  • Must be parallel to camera (although can be rotated)
  • Too large: small range of depth to be in range
  • Cannot find 6-DOF (degree of freedom)
Fiducial Marker Processing Stages

- identifying feature (unique feature)
- verification and identification (is it a marker, if so which one?)

Unique Feature Detection

- binary image -> morphology (ARToolkit, Intersense, Matrix, BSM, Cybercode)
- edge-based: find unique feature from edges (ARTag)

Processing

- correlation (ARToolkit)
- topological (ReacTIVision)
- symbolic digital (Intersense, Cybercode, Canon, BSM, Matrix, SCR, HOM, ARTag)
ARToolkit Fiducial Marker System

- Developed by Dr. Hirokazu Kato (Osaka University)
- Internationally popular – used in AR, HCI projects
- Freely downloadable

Must load marker file and correlate for every marker to be detected.

False Positives.

ARToolkit Drawbacks:
- False Detection - marker reported that doesn’t exist.
- Inter-marker confusion - wrong marker ID reported.
- Must set c.f. threshold

ARToolkit reports markers with a Confidence Factor $0 < c.f. < 1$
ARTag Fiducial Marker System

- Digital (symbol based)
- Bi-tonal (only black and white) patterns
- Edge-based unique feature detection – no threshold required
- 2002 markers – no pattern files
- Digital Methods: Error Correction, CRC-16 Checksum
ARTag

2002 possible markers

Creating Markers: Encoding codes

Detecting Markers: Identification and decoding
False Detection Probability

- 2002 possible markers = 1001 different internal codes
- x4 rotations = 4092 codes out of $2^{36} = 68.7$ billion
- 1023 codes $\times (1+36+36^2) \times 4 = 5.45$ million correctible interiors
- False detection rate = $5.45 \times 10^6 / 6.87 \times 10^{10} = \%0.0034 = \text{approx. } 1/30,000$

Inter-marker Confusion Probability

- Examine Hamming distances
- Probability broken into 2 components: situation and code dependent.
  Hamming distance is coeffs for latter.
Fiducial Marker System Design: Hamming Distance between Patterns

**Hamming Distance: Definition** = # of different bits

\[10010 \text{ and } 10001 \quad \text{H.D.} = 2\]

<table>
<thead>
<tr>
<th>Pattern A</th>
<th>Pattern B</th>
<th>Hamming Distance between Patterns A and B = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 1 0 1</td>
<td>1 0 0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>1 1 1 1 1 1</td>
<td>1 1 0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>1 0 1 1 0 1</td>
<td>1 0 1 1 0 1</td>
<td></td>
</tr>
<tr>
<td>0 0 1 1 0 0</td>
<td>0 0 1 1 0 0</td>
<td></td>
</tr>
<tr>
<td>1 1 0 0 1 1</td>
<td>1 0 0 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>
ARTag – Hamming Distance between Patterns

\[
P(\neq A) = \sum_{n=1}^{36} HD(n) \cdot p(n)q(36-n)
\]

\[
P(\neq A) = \sum_{n=1}^{36} HD(n) \cdot p^n(1-p)^{36-n}
\]
Recommended Marker Order

```c
artag_recommended[ 0] =  4;  artag_recommended_hd[ 0] = 36;
artag_recommended[ 1] = 57;  artag_recommended_hd[ 1] = 16;
artag_recommended[ 8] = 397; artag_recommended_hd[ 8] = 14;
```

Use global variables `artag_recommended[0…1000]` - or - `id=artagLoadPatt(char *patt_name);`
Comparing ARTag and ARToolkit, ARToolkit Plus

*Lighting immunity*

ARToolkit Plus: thresholding/binary morphology based
Comparing ARTag and ARToolkit, ARToolkit Plus

Occlusion immunity
Comparing ARTag and ARToolkit Plus

Inter-marker confusion rate (how often is marker ID wrong)
Comparing ARTag and ARToolkit Plus

Inter-marker confusion rate (how often is marker ID wrong)
Comparing ARTag and ARToolkit Plus

*Inter-marker confusion rate* (how often is marker ID wrong)
Comparing ARTag and ARToolkit Plus

*False Negative rate* (how often is marker missed)
Symbolic: digital codes in markers

- error checking – verification - checksums – false positive rate
- redundancy – recover from falsely digitized bits - FEC – false negative rate
- digitization process – pixels to digital ‘1’ & ‘0’ – bitonal = no photometric cal.
- code size – how many bits encoded
  - broken into ID, checksum, FEC
  - library size & minimum image size
- code bits lost to orientation determination

Code Size, Error checking, Redundancy

- Intersense: 15 bits, none, none
- BSM: 16 bits: 4 bits orientation, 7 bits ID, 6 bits parity
- Matrix: 25 bits, some allocated to checksum, none
- Canon: 9 bits: 4 bits orientation, 5 bits ID, none, none
- ARVIKA (IGD, HOM, SCR): 16, 20 bits, ? ID, ?, ?
- ARTag: 36 bits: 10 bits ID, 16 bits checksum, 10 bits FEC
Use ARTAG: Rev2 functions

```
#include "artag_rev2.h"

Initialize ARTag
init_artag(xsize,ysize,num_bytes_per_pixel);

Load array file and assign arrays, markers
load_array_file("panel_set.cf");
artag_object_id=artag_associate_array("base0");

Process video frame
artag_find_objects(cam_image,rgb_greybar);

Set OpenGL modelview matrix from detected objects
if(artag_is_object_found(artag_object_id))
{
  float opengl_matrix[16];
  artag_get_object_opengl_matrix(artag_object_id,opengl_matrix);
  glLoadMatrixf(opengl_matrix);
  glBegin(GL_TRIANGLES);
  glColor3f(1, 0, 0);
  ...

Link in <artag_rev2.lib>
```
Use ARTAG: Rev2 functions

**Initialization**

- `init_artag()`
- `load_array_file()`
- `artag_associate_array()`
- `artag_associate_marker()`
- `atag_get_object_coordframe_range()`

**Every Video Frame**

- `artag_find_objects()`
- `artag_is_object_found()`
  - `artag_get_object_opengl_matrix()`
  - `artag_project_point(artag_object_id,x,y,z,&u,&v)`
  - `artag_project_between_objects(id1,scale,id2,x1,y1,z1,&x2,&y2,&z2)`
  - `artag_get_object_homography(artag_object_id,&h11,&h12,…h33)`

**Rev1 functions – direct marker detection**

- `artag_find_marker()`
- `artagDetectMarker()`
- `artag_create_marker()`
- `artag_get_id()`
Use ARTAG: Rev1 functions

#include "artag_rev2.h"

Initialize ARTag

init_artag(xsize,ysize,num_bytes_per_pixel);

Give image unsigned char* to ARTag

error=artag_find_marker(image,rgb_greybar,
    artag_pattern_id,
    artag_pattern_x0,artag_pattern_y0,
    artag_pattern_x1,artag_pattern_y1,
    artag_pattern_x2,artag_pattern_y2,
    artag_pattern_x3,artag_pattern_y3,
    &num_markers_found);

Link in <artag_rev2.lib>

Use global variables artag_recommended[0...1000]
ARTag Drop-In Functions

Try out ARTag in your existing ARToolkit programs.

```cpp
#include "artag_rev2.h"

Initialize ARTag

init_artag(xsize,ysize,num_bytes_per_pixel);

Replace arDetectMarker() with artagDetectMarker()

-optionally:
Replace arLoadPatt() with artagLoadPatt()

Link in <artag_rev2.lib>
DEMO New SDK (ARTag Rev2) : 2D augmentation

“2d_augmentation” demo app (with source) from download www.artag.net
DEMO New SDK (ARTag Rev2): 3D augmentation

“3d_augmentation” demo app (with source) from download www.artag.net
DEMO New SDK (ARTag Rev2) : 3D augmentation

“3d_augmentation” demo app (with source) from download www.artag.net
DEMO New SDK (ARTag Rev2) : transferring X,Y,Z

“artag_cad” demo app (with source) from download  www.artag.net
ARTag Rev2 Fiducial Marker System: Vision-based Tracking for AR

The END – thank you for listening

Mark Fiala
Computational Video Group
Institute of Information Technology
National Research Council
mark.fiala@nrc-cnrc.gc.ca

www.artag.net