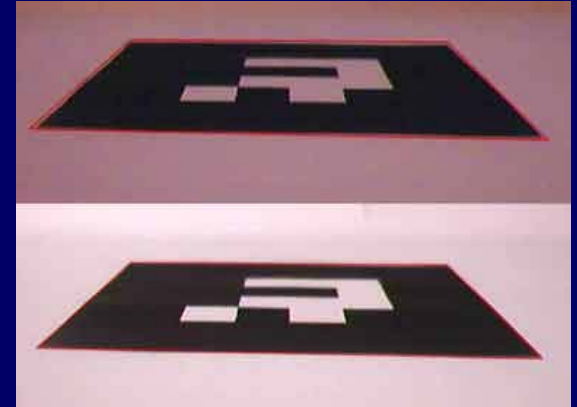


# A Lightweight ID-Based Extension for Marker Tracking Systems



---

Daniel Flohr<sup>1</sup>, **Jan Fischer**<sup>2</sup>

<sup>1</sup> WSI / GRIS, University of Tübingen, Germany

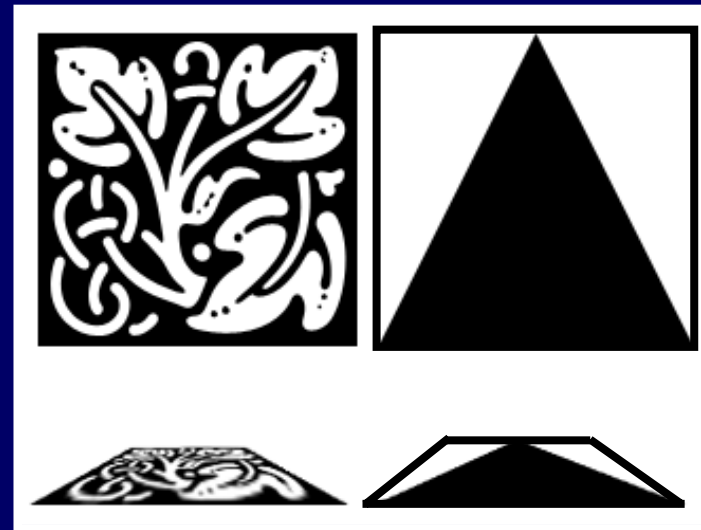
<sup>2</sup> Island Graphics Group, University of Victoria, Canada

IPT - EGVE 2007

13th Eurographics Symposium on Virtual Environments  
10th Immersive Projection Technology Workshop

# Motivation

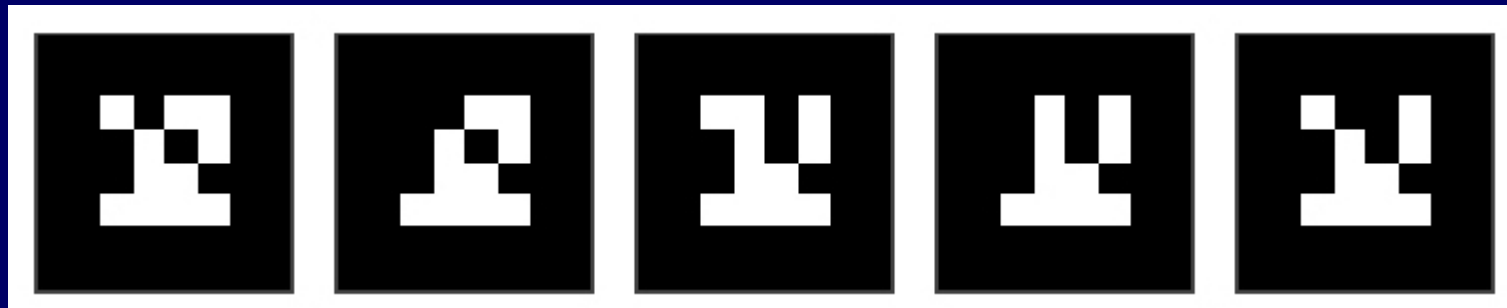
- Image-based marker tracking a **central component** of many AR applications
- Widespread technology: **ARToolKit**
- Some applications require **many markers**



- Manual creation / capturing of markers **tedious**
- **Marker design** impacts tracking performance
- Errors introduced by **capturing**

# ID-based Markers

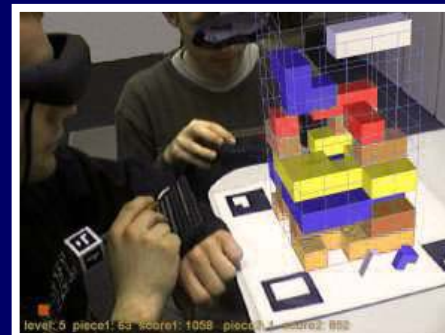
- **Automatic** generation of marker patterns
- Large, **monochrome** (binary) patterns
- **No capturing** process necessary



- Patterns generated / identified based on (integer) **ID number**

# Related Work

- **ARTag** [Fiala 2004]
- **ARToolKit Plus**  
[Wagner, Schmalstieg 2007]
- **ARToolKit Professional**  
([www.artoolworks.com](http://www.artoolworks.com))



# Objective



- Existing systems:
  - Typically **separate** libraries or **patches** to the ARToolKit code
- Can lead to **complex integration** in existing applications
- Proposal: **Lightweight, self-contained** addition to ARToolKit applications
  - Currently only three C++ files
  - **Minimal changes** to existing code

# Outline

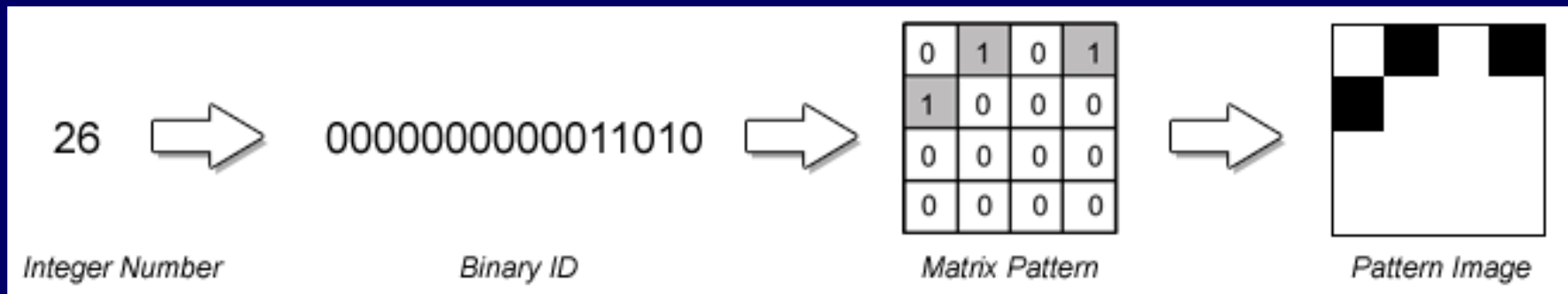
---



- Introduction
- **ID-based marker generation**
- Evaluation
- Example Application
- Conclusions

# Marker Generation (1)

- Monochrome **4 x 4 matrices**
- Corresponding to 16-digit **bit string**

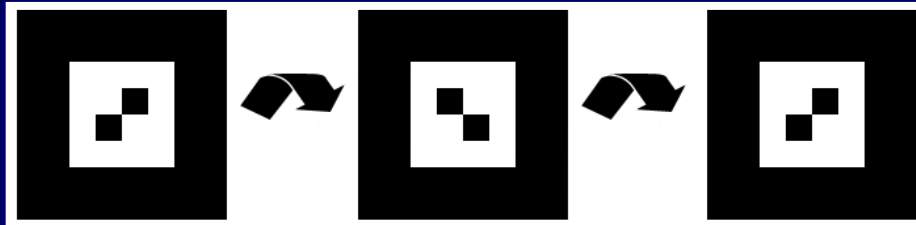


- Marker patterns stored in **temporary files**
- Corresponding PBM files for **printing patterns**

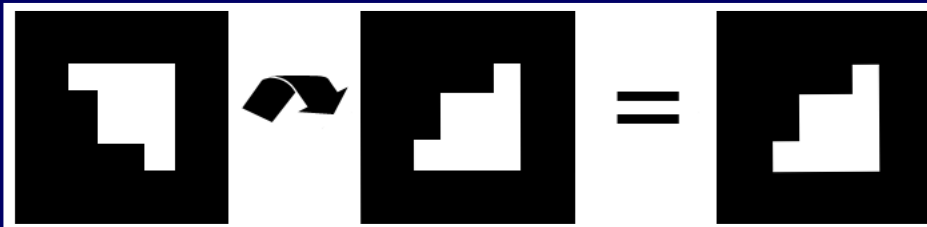
# Marker Generation (2)

- Valid patterns:

- Test for **self-similarity**



- Comparison with **(rotated) existing** patterns



- Ca. **16.000** valid patterns (valid IDs **precomputed**)
- For simpler usage:  
**Continuous integer ID** instead of bit-string

# Usage



```
OBJECT_T object[3];

for( i = 0; i < 3; i++ ) {
    object[i] = binaryIDHandler->getObject_T(i, 40.0f);

    // Generate temporary pattern file
    binaryIDHandler->createARToolkitPattern(i, object[i].patt_name);

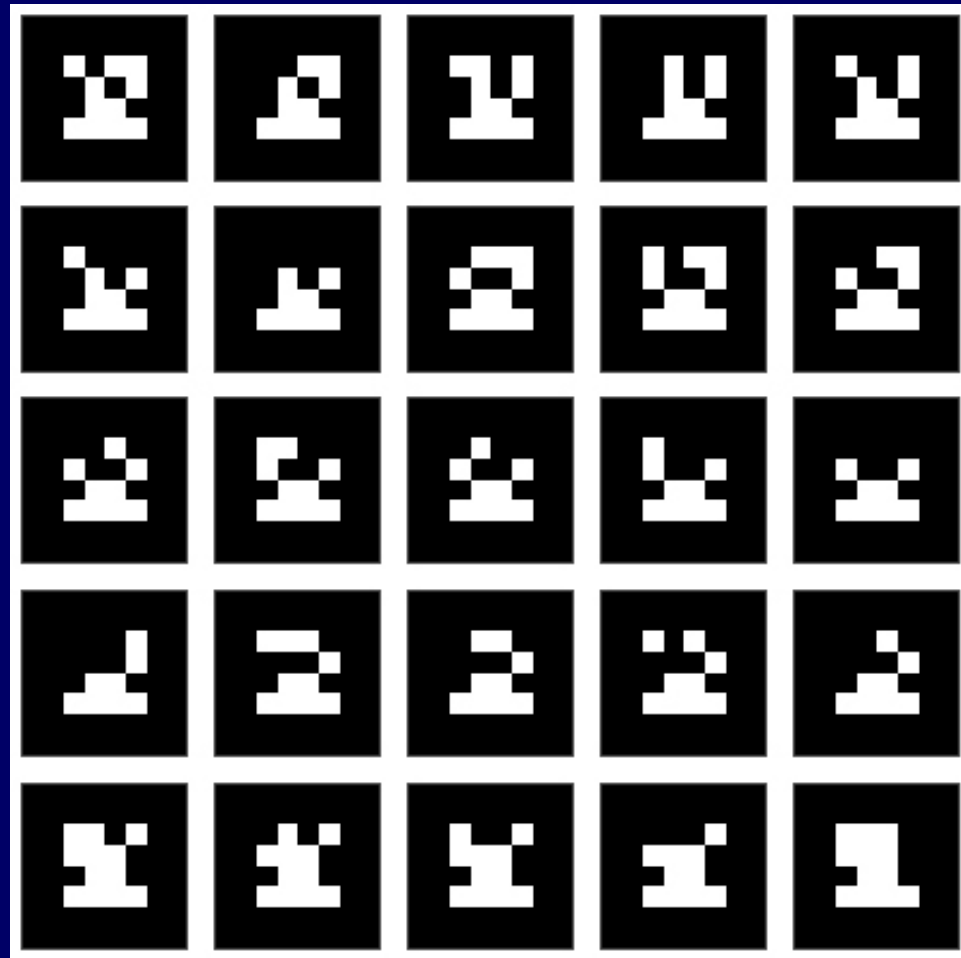
    // Standard ARToolkit pattern loading call
    object[i].patt_id = arLoadPatt(object[i].patt_name);

    // Store mapping from binary pattern ID to ARToolkit ID
    binaryIDHandler->mapARtoBin(i, object[i].patt_id);
}

:
:

// Look up binary ID for recognized ARToolkit ID
int binID = binaryIDHandler->getBinaryID(recognizedARTkID);
```

# Example Patterns



# Outline

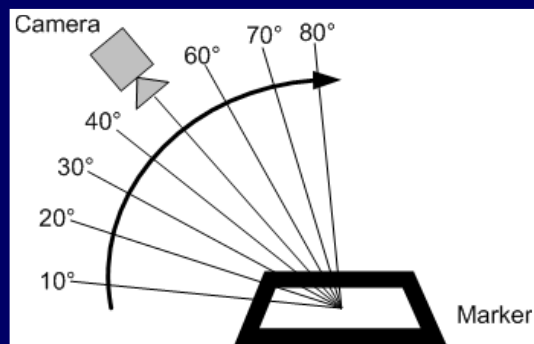
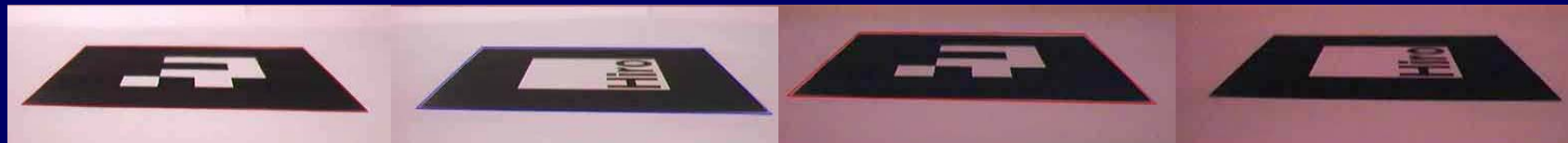
---



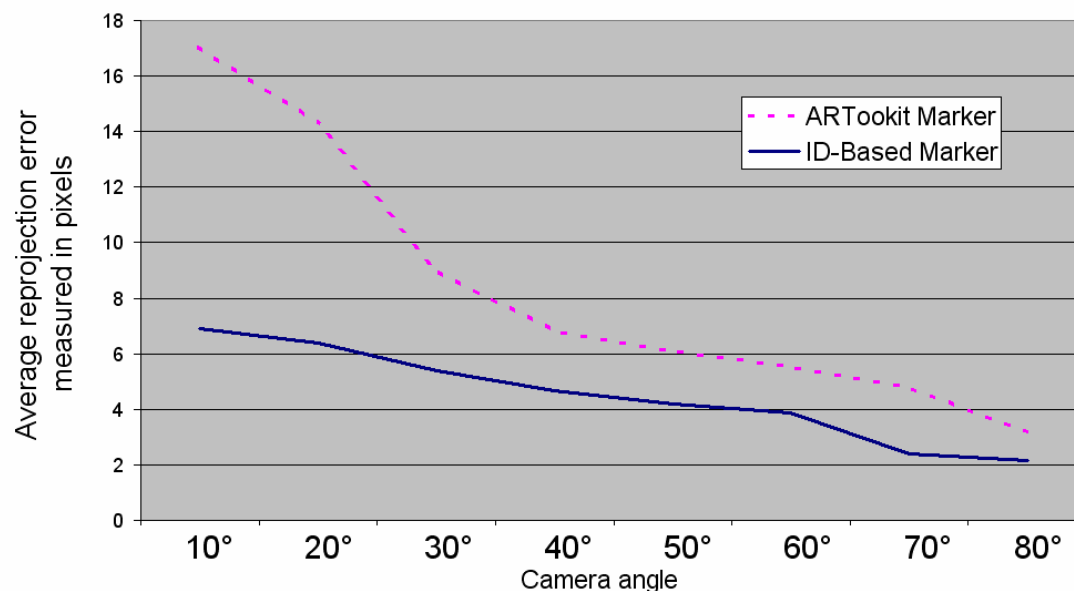
- Introduction
- ID-based marker generation
- **Evaluation**
- Example Application
- Conclusions

# Bad Lighting Conditions

- Comparison with **typical manually defined** markers

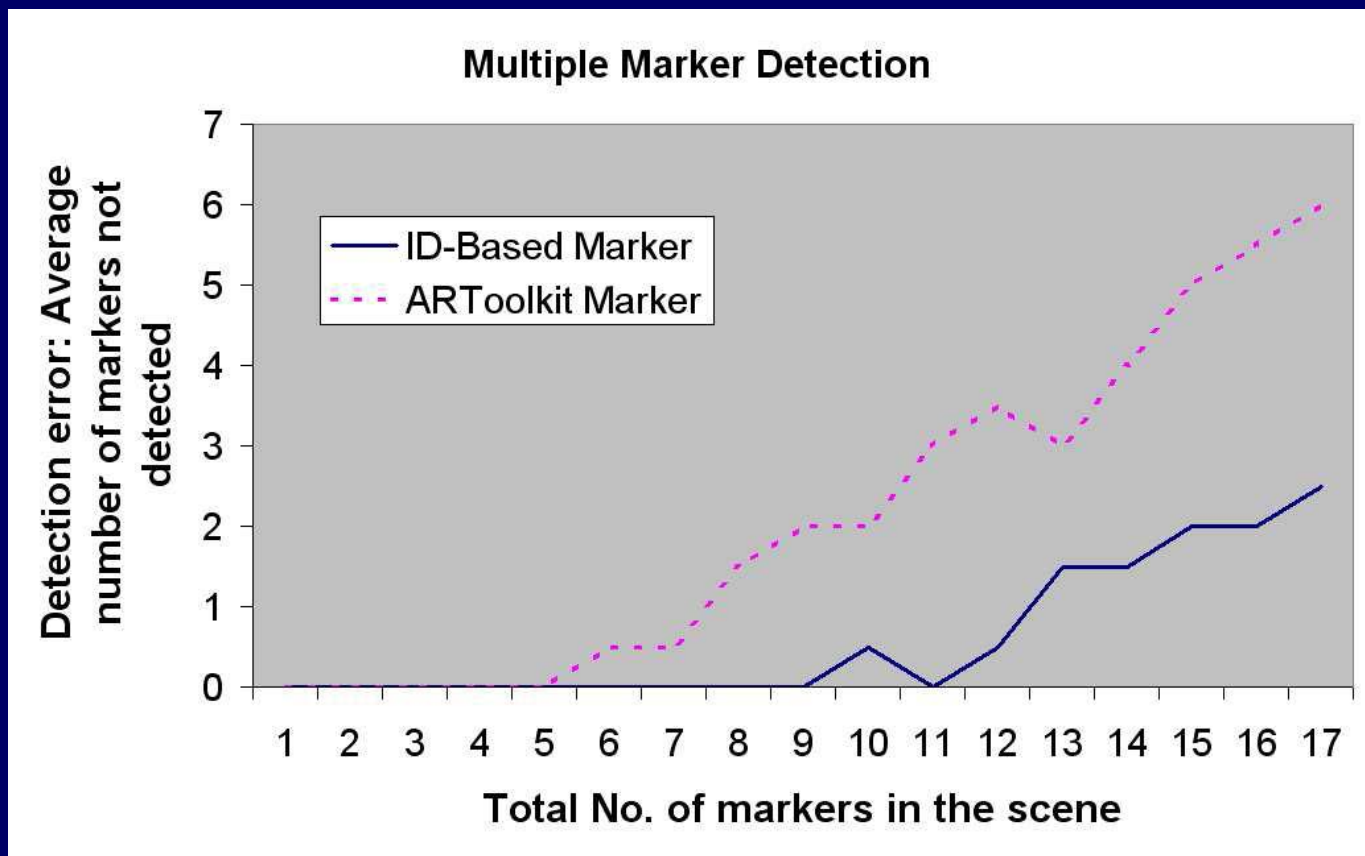


Reprojection error under bad lighting conditions and varying camera angles



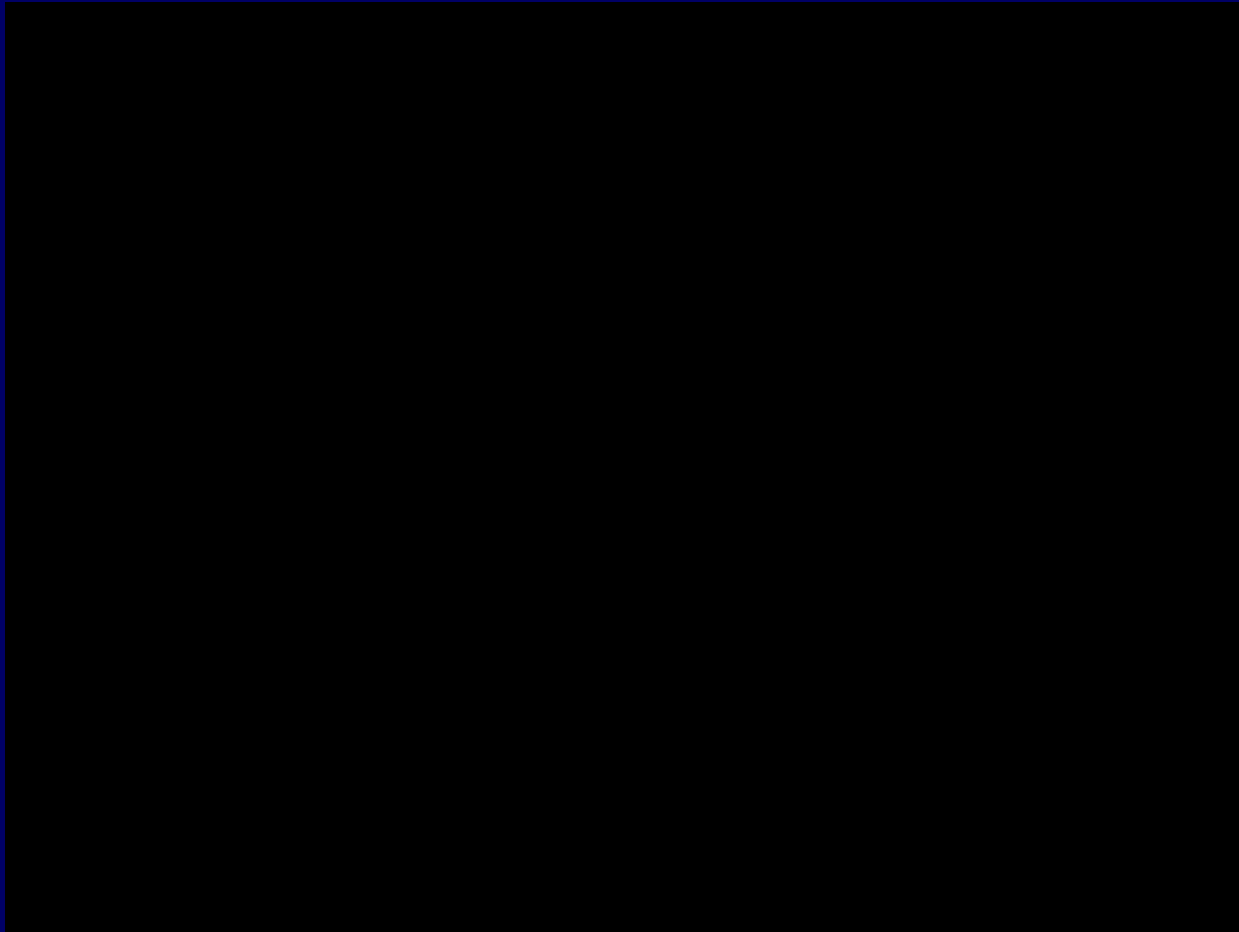
# Multiple Markers

- Setup with **multiple markers** on a base plate



# Evaluation - Video

---

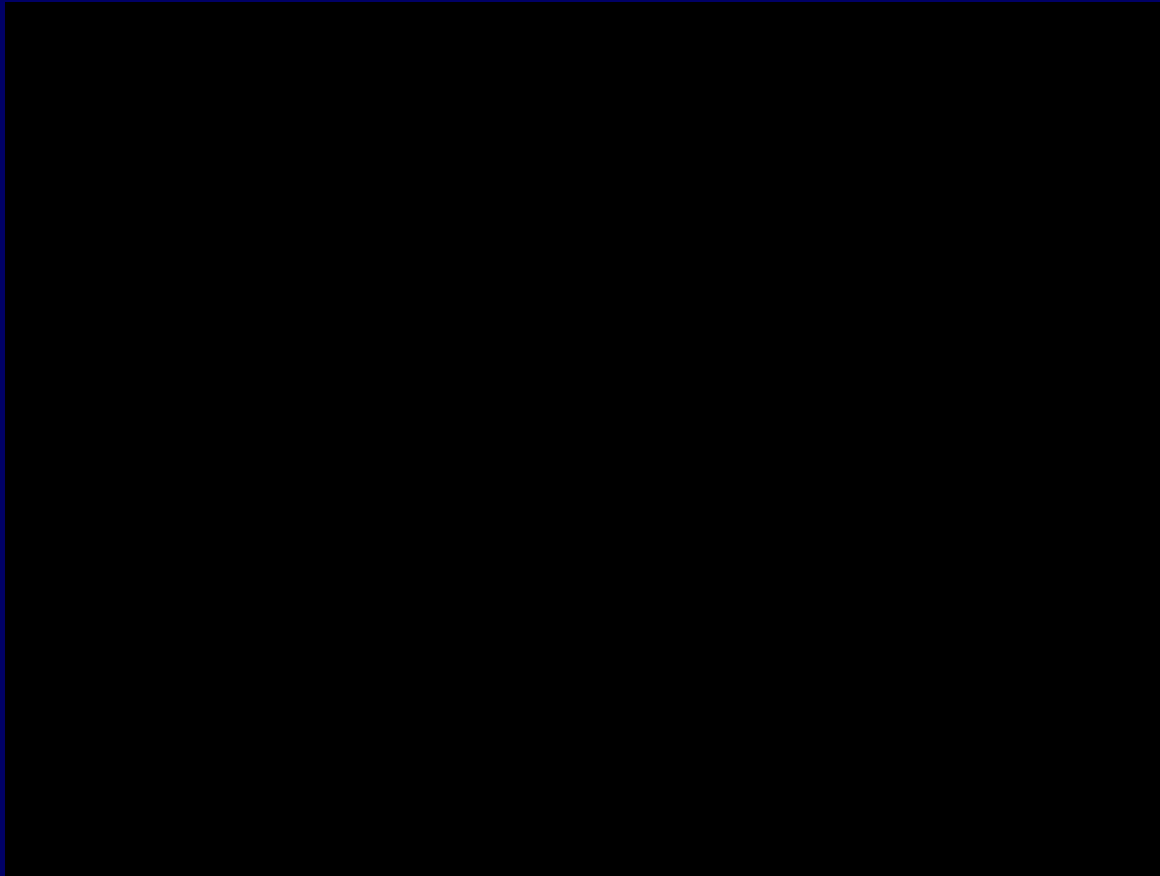


# Example Application

---



- **Tangible interaction** application, requires many markers



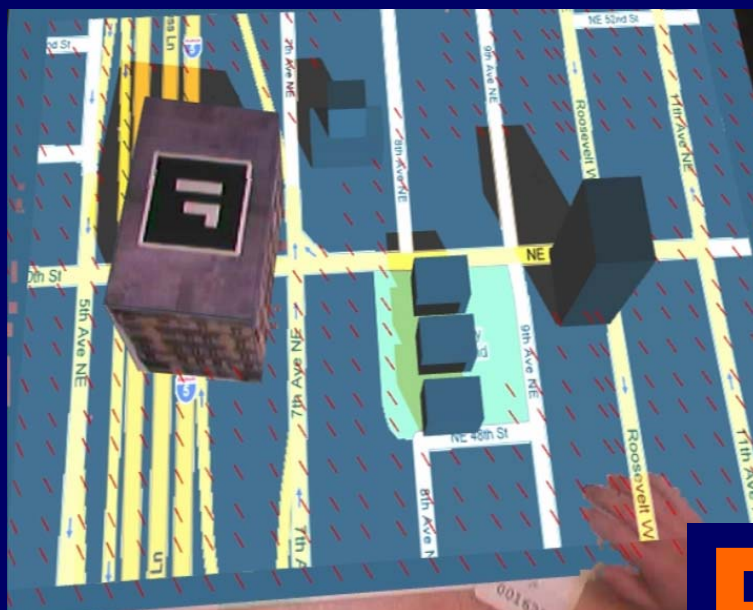
# Conclusions

---



- **Lightweight, self-contained** ID-based marker tracking extension
- Can improve **tracking accuracy** in certain situations
- Drawback: Cannot overcome basic **ARToolKit limitations**
  - Sensitivity to **occlusion**
  - Sensitivity to **extreme distances / angles**
  - Sensitivity to camera **image edges**

# END



# END

Acknowledgments: Deutsche  
Forschungsgemeinschaft

**DFG**