Real-time 3D Tracking with Camera Phones

Daniel Wagner
Augmented Reality on Mobile Phones

- Low cost, widely spread platform
  - Billions of phones deployed
  - People know how to use them
  - Strong demand from commercial side
  - Huge chance for AR!

- Target practical applications
  - Easy to use
  - 15-30 Hz overall frame rate
  - Robust tracking
Mobile Phone Camera Tracking

Tracking for Augmented Reality always means **Pose Tracking** (6DOF)

- Optical Flow
  - Very simple, but does not give a pose
- Marker Tracking
  - Works well, but limited in its applications
  - Hardly a research topic anymore
- Natural Feature Tracking
  - Currently a hot topic!
History of AR Tracking on Phones (1)

- 2003
  - ARToolKit on PDA
  - Wagner et al.

- 2004
  - 3D Marker on Phone
  - Möhring et al.

- 2005
  - ARToolKit on Symbian
  - Henrysson et al.
History of AR Tracking on Phones (2)

- 2005
  - Visual Codes
  - Rohs et al.

- 2006
  - Studierstube Tracker
  - Wagner et al.

- 2007
  - WikEye
  - Schöning et al.
History of AR Tracking on Phones (3)

- 2008
  - Advanced Marker Tracking
  - Wagner et al.

- 2008
  - Natural Feature Tracking
  - Wagner et al.

- 2009
  - High speed Natural Feature Tracking
CPU/Memory Limitations of Mobile Phones

- Small memory
  - Even though phones today have 64-128MB RAM, consider 2-5 Megabytes as maximum

- Weak processing power
  - 200-600 MHz, Single core
  - Typically no FPU (floating point ~40x slower than integer)
  - Slow memory access, small caches

→ Code optimized for phones runs **5-10x slower** on a high-end phone than on an average PC (2GHz, single core)

→ Not going to change quickly due to battery power
Tracking by Detection

- This is what most „trackers“ do...
- Targets are detected every frame
- Popular because tracking and detection are solved simultaneously
Natural Feature Tracking by Detection

**SIFT**
- State of the art for object recognition
- Known to be slow (best implementation for phones is ~10-100x too slow for real-time use)
- Typically used off-line

**Ferns**
- State of the art for fast pose tracking
- Known to be memory intensive (requires ~10x too much memory for phones)
- Long training phase

Performance of SIFT and Ferns modified for mobile phone tracking

![Bar chart showing performance comparison between SIFT and Ferns for different devices and configurations.]

- **Overall tracking time**
  - **0ms to 100ms**
  - PC, iPAQ, N95 Float, N95 Fixed, Moto Q9

- **FERNS Ad**
- **FERNS Cars**
- **FERNS Vienna**
- **SIFT Ad**
- **SIFT Cars**
- **SIFT Vienna**
NFT with SIFT on a Mobile Phone
Doing it better: Dedicated Detection and Tracking

Target Recognition System

Activate recognition system if tracking was lost

Target Tracking System

Activate tracking system if target was found
Detection and Tracking
High Speed Tracking on the Mobile Phone

**Robust High Speed Natural Feature Tracking**

**Tracking of Multiple Targets**

**Memory Requirements:**

~300KB/Target

**Device:** ASUS P552W, 624MHz

**Rendering:** OpenGL ES 1.1

**Camera:** 320x240, 20Hz
Workflow of our Tracker

- During startup: find features in a reference image
- At runtime:
  - Take previous pose and apply motion model
    - Get estimate for what we are looking for
  - Create affine warped patches of reference features
    - Closely resemble how the feature should look in the camera image
  - Project patches into camera image and use normalized cross correlation (NCC) to match
PatchTracker in Action (1)
PatchTracker in Action (2)
How fast is it?

Performance on the phone:
- PhonySIFT: 38.3 ms
- PhonyFerns: 41.3 ms
- PhonySIFT with PatchTracker: 8.4 ms
- PhonyFerns with PatchTracker: 8.3 ms

Performance on the PC:
- PhonySIFT: 3.8 ms
- PhonyFerns: 3.2 ms
- PhonySIFT with PatchTracker: 1.0 ms
- PhonyFerns with PatchTracker: 1.0 ms
## Orthogonal Strengths and Weaknesses

<table>
<thead>
<tr>
<th></th>
<th>SIFT/Ferns</th>
<th>PatchTracker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognize many targets</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Detect target</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Initialize tracking</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Speed</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Robust to blur</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Robust to tilt</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td>Robust to lighting changes</td>
<td>✗</td>
<td>✔️</td>
</tr>
</tbody>
</table>
More Results... (1)

Mobile Phone Augmented Reality

at 30 Frames per Second

using Natural Feature Tracking

(all processing and rendering done in software)
More Results... (2)
Conclusions & Future Work

- NFT sometimes more robust than markers
  - Bad lighting (blur)
  - Occlusions
- Many open issues
  - Non-planar targets
  - Large targets (rooms, building, cities)
  - Automatic target acquisition (SLAM)
  - GPU implementations
Thank you for listening...