Real Time Augmented Reality on Mobile Phone

Fisher
Project Goal

• Explore the algorithms in real time tacking
• Build a system for visual based augmented reality
Application

- Augmented Game
- Interactive and Personalized Annotation
- Design Preview
- Face Tagging
- ...

Challenge

- Algorithm tradeoff between speed and accuracy
- Performance of cell phone
- Tweak the system to make it work
- Hardware limitation
Previous Work

• Marker Tracking
  – Well Studied
  – Fast and Accurate
Previous Work

• Model Based Tracking
  – Look better
  – Feasible as the rapid development of mobile device
  – Different approaches
Previous Work

• Model Based Tracking
Previous Work

• SFM Based
  – Computational Expensive
  – Low Accuracy

• Example
  – Real-time SFM
  – Mono SLAM
Hybrid System

- Everything is on client
  - No delay from networking
  - Need to be simple

- Detect then track
  - Give a hint in the scene – Image template
  - Use feature tracking to locate the object roughly
  - Use texture to track the changes between frames
Hybrid System

Initial → Detection → Tracking

Detection → Detect

Detect → Lose
Detecting

• Not a big deal
  – Extract features (SIFT, SURF, FERNS)
  – Match features based on threshold
  – Use RANSAC to get homography
Tracking

• Use texture based minimization algorithm
  – Traditionally, Newton method is used
  – High convergence rate – Quadratic
  – Hessian matrix is needed
  – Not stable
Tracking

• Several first-order approximations are purposed
  – Gradient descent
  – Gauss-Newton
  – Levenberg-Marquardt

• However, low convergent rate
Tracking

• In this work, another approximation is used
  – Called ESM
  – Efficient second-order minimization
  – Only the Jacobian matrix is needed
  – Faster
  – More stable
  – It has been commercialized.
Tracking

\[ f(x) = \frac{1}{2} \|y(0) + J_{esm}x\|^2. \]

\[ x_0 = J_{esm}^+ y(0) \]

- \( y \): the difference between image template and hypothesis
- \( x \): the point coordinates
Tracking

• Lie Algebra
  – Simplify the calculation of Jacobian matrices
  – Nice feature on differentiation
  – However, matrix exponential is needed.
Tracking

Matrix Exponential
• Taylor Series
• The Scaling and Squaring method
• Pad´e Matrix Approximation

Mat X = A.clone();
double c = 1.0 / 2;
Mat E = I + c * p;
Mat D = I - c * p;
int q = 6;

for (int k = 2; k <= q; k++) {
c = (double)c * (q - k + 1) / (k * (2 * q - k +1));
X = A * X;
E = E + c * X;
if (k & 1)
    D = D - c * X;
else
    D = D + c * X;
}
Detection
Tracking
Tracking
## Performance

<table>
<thead>
<tr>
<th>Template Size</th>
<th>128 by 128</th>
<th>256 by 256</th>
<th>521 by 512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection (SURF)</td>
<td>100ms</td>
<td>210ms</td>
<td>229ms</td>
</tr>
<tr>
<td>Tracking (6 loops)</td>
<td>30ms</td>
<td>128ms</td>
<td>550ms</td>
</tr>
</tbody>
</table>

- Test image size: 720 by 480
- Run on desktop
Running on iPhone

• First attempt using private framework
  – Look feasible but riskier

• Switch to some other framework
  – Start over but the previous experience helps a lot
Conclusion

• An augmented reality system is built
• The tracking algorithm is not very stable
• Hard to deal with blurred images and drift
  – Mobile devices are good at these
• 3D object tracking
• Expect to work without hint