Real-Time Detection and Tracking for Augmented Reality on Mobile Phones

D. Wagner, A. Mulloni and D. Schmalstieg

Presented by

Yuansi Chen and Lingqi Yan

Outline

1. Motivation and Related Work
2. Modified Features Detectors
3. Performance and Analysis

Motivation

Motivation

- Limited computational resources (speed and memory) on Mobile devices
- Natural feature tracking infeasible: SIFT and Ferns

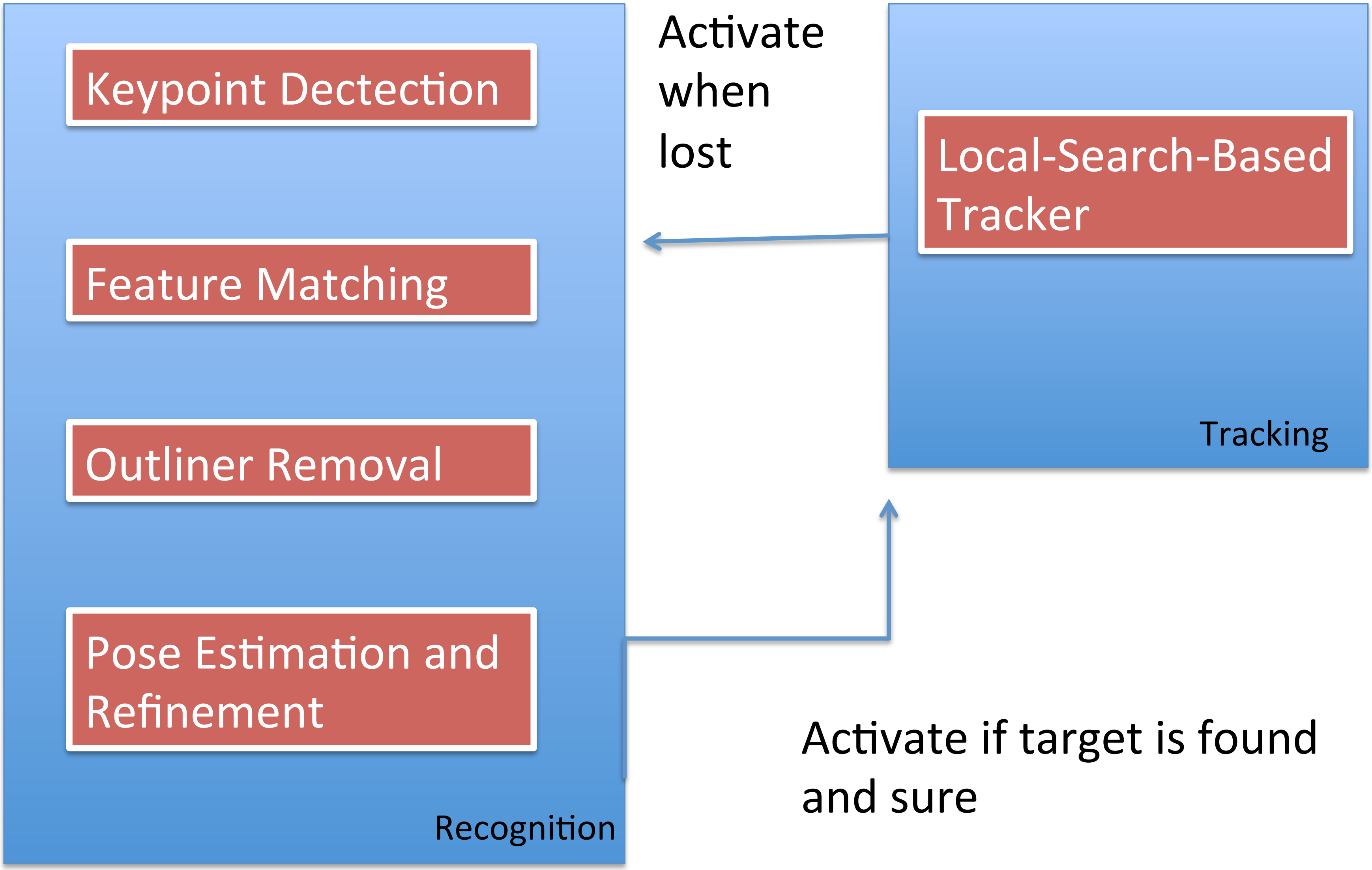
Goal

- Enough speed improvement for real-time AR processing
- with Limited memory
- without losing too much quality
- on real phones (<33ms/frame)

Related Works

- General Feature Detectors for PCs (slow)
- Outsource the tracking task to PCs via wifi. (AR-PDA project: 10s per frame is still slow)
- Marker tracking: restricted applications

Detection and Tracking Routine



Keypoint Dectection

Feature Matching

Outliner Removal

Pose Estimation and Refinement

Recognition

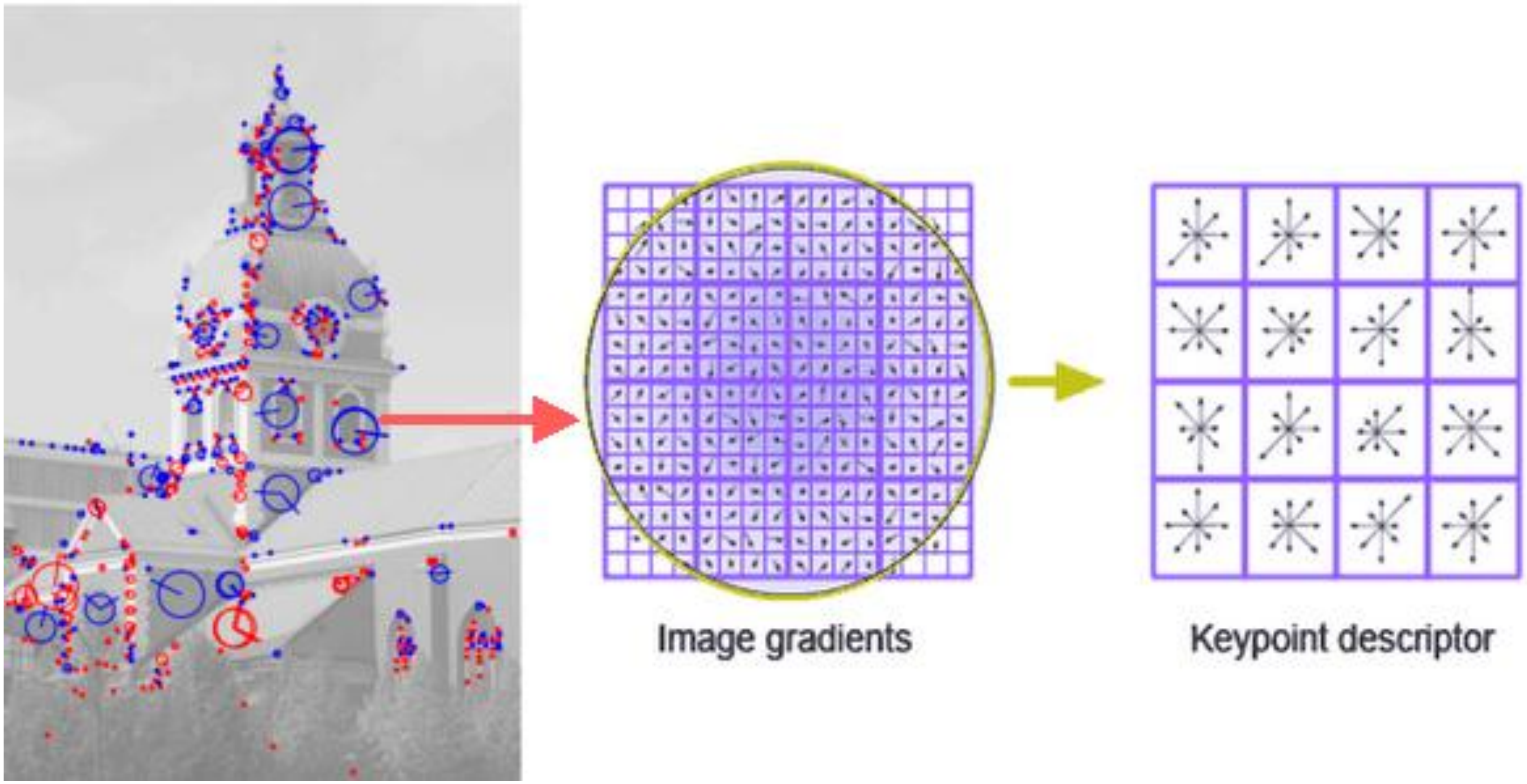
Activate when lost

Local-Search-Based Tracker

Tracking

Activate if target is found and sure

Scale Invariant Feature Transform (SIFT)



Ferns

- Feature detection as classification
- Binary Feature F(p)
- C = argmax P(Ci|F)
- Instead of storing full joint distribution, add independence:

  P(F|C) = ΠP(F_S|C)

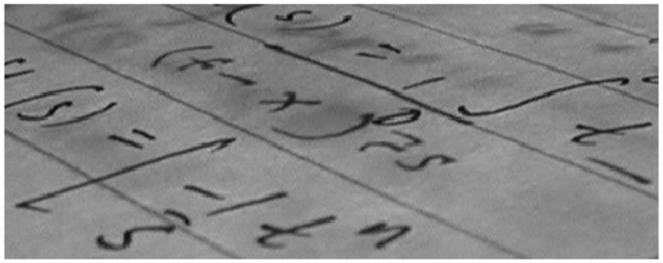
FAST Corner Detector

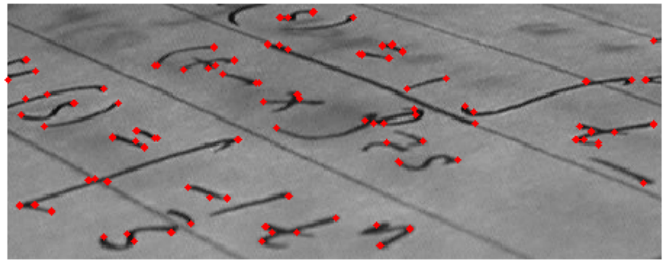
Ref from:

<Machine learning for high-speed corner detection>

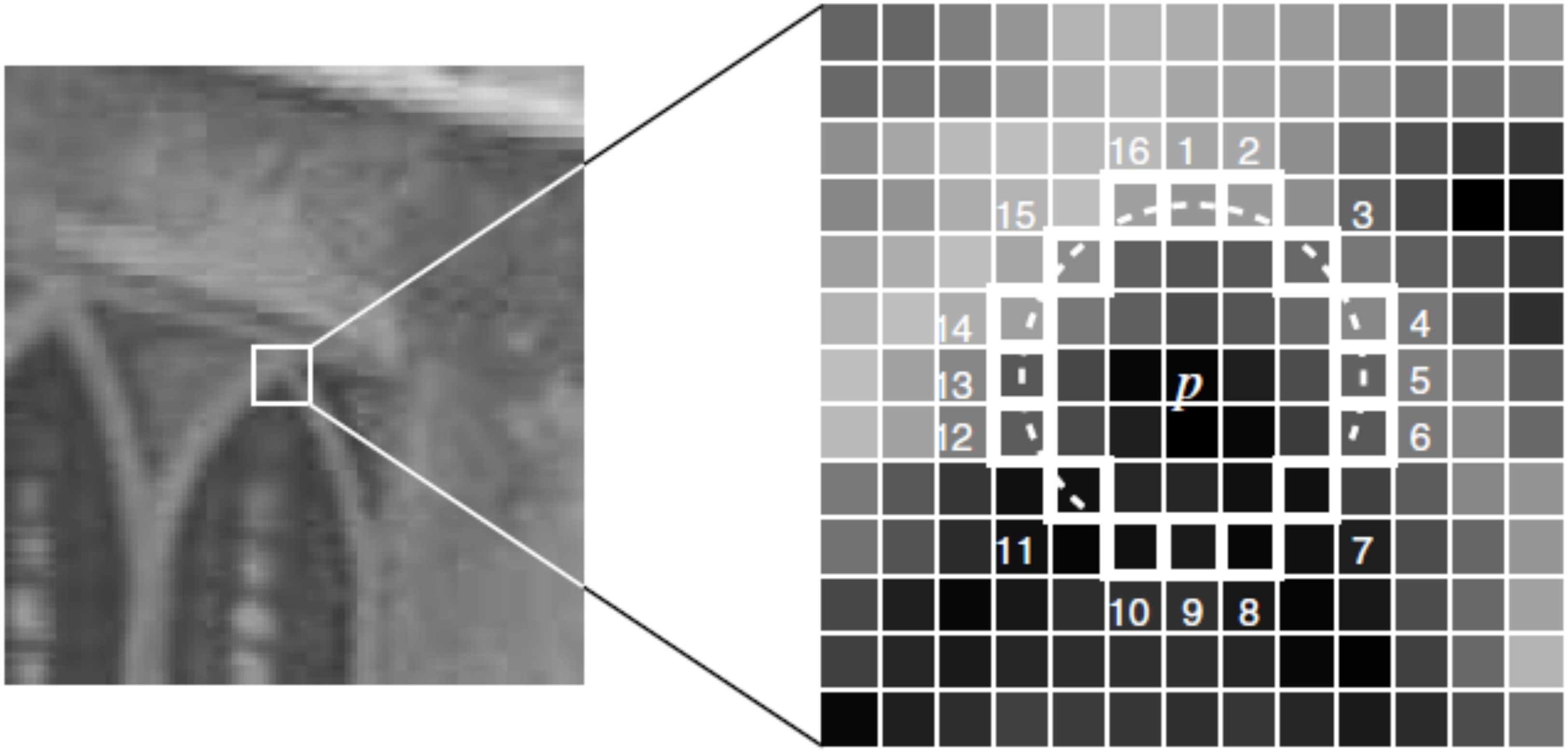
By Edward Rosten and Tom Drummond, University of Cambridge

- **Features from accelerated segment test (FAST)**
- A corner detector many times faster than DoG but not very robust to the presence of noise
- Based on intensity level tests





FAST Corner Detector



SIFT to PhonySIFT

Main Modifications:

- Uses FAST corner detector to all scaled images to detect feature points instead of scale-crossing DoG
- Only 3x3 subregions, 4bins each , creates 36-d vector
- Using a Spill tree

Ferns to PhonyFerns

Main Modifications

- Uses FAST detector to increase detection speed
- Reduces each ferns size
- Uses 8-bit size to store probability instead of using 4 bytes float point value
- modifying the training scheme to use all FAST responses within the 8-neighborhood

Outliner Removal

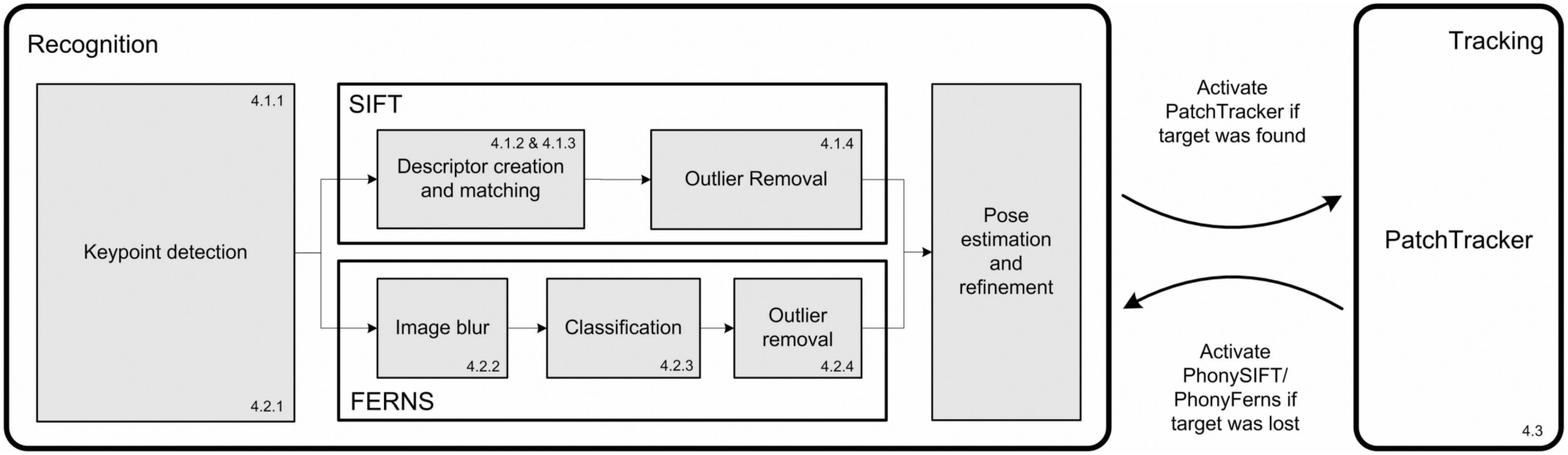
- Orientation Estimation
- Homography verification based on RANSAC/ PROSAC

PatchTracker

Ideas:

1. Both the scene and the camera pose change only slightly between two successive frames
2. New feature positions can be successfully predicted by old one with defined range search

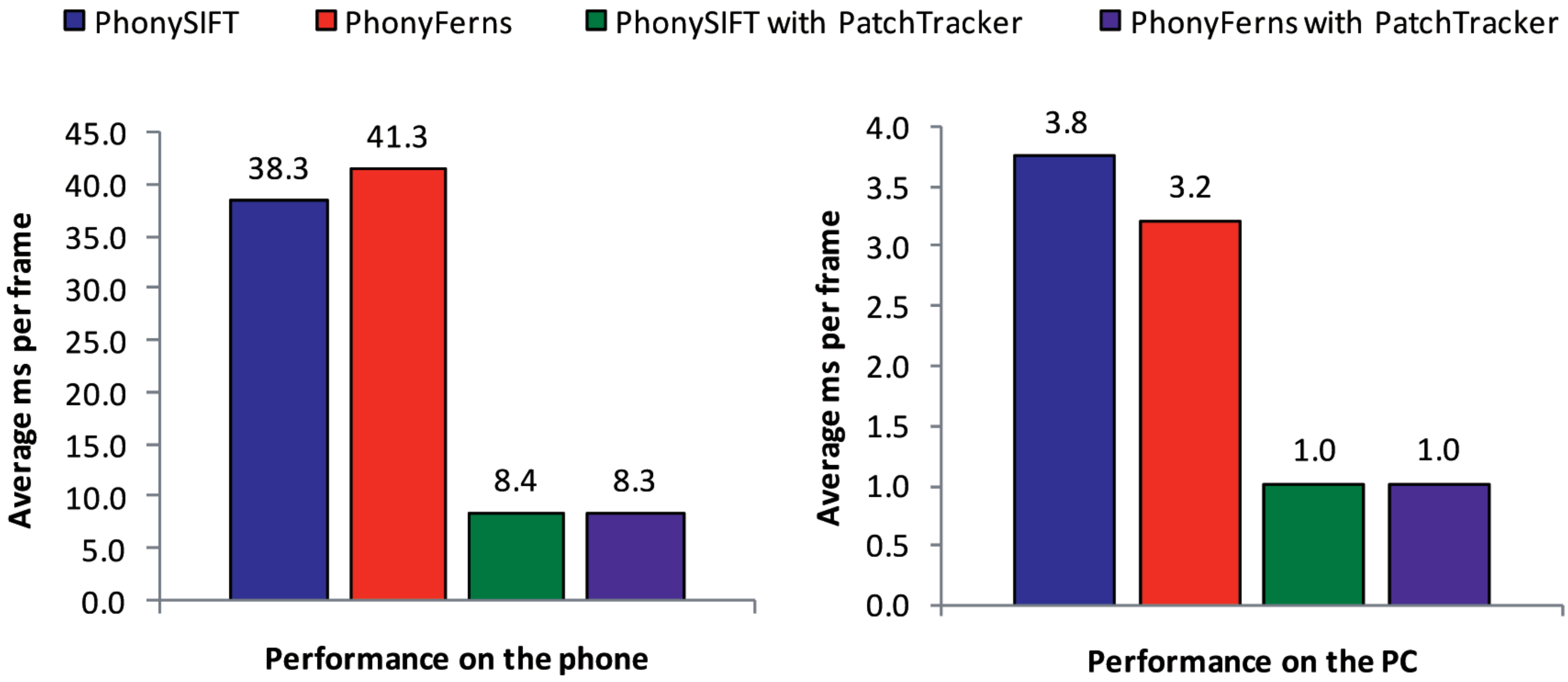
Combined Tracking



Performance & Analysis

- Platform:Asus P552W (Cellphone)
  - 624Mhz CPU
  - 240x320 screen resolution
  - No float point unit
  - No 3D acceleration
- Platform:Dell Notebook (PC)
  - 2.5Ghz , limited to use single core
  - With float point support

Speed



🟦 PhonySIFT
🟥 PhonyFerns
🟩 PhonySIFT with PatchTracker
🟪 PhonyFerns with PatchTracker
Average ms per frame
45.0
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
38.3
41.3
8.4
8.3
Performance on the phone
Average ms per frame
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
3.8
3.2
1.0
1.0
Performance on the PC

Robustness over different objects

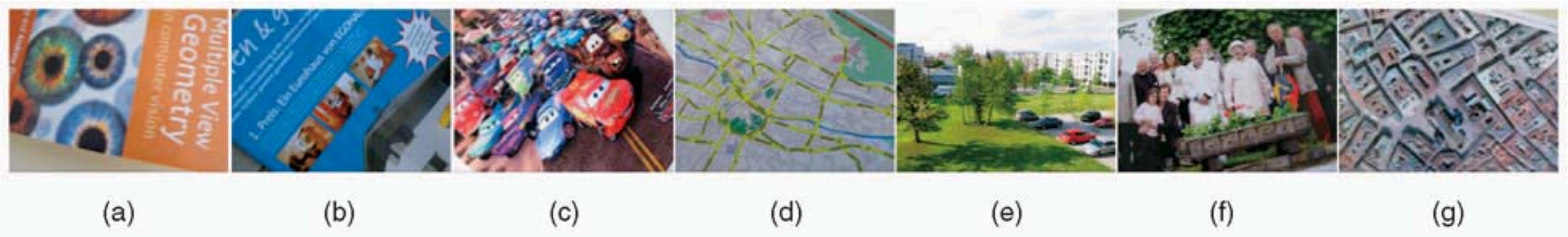
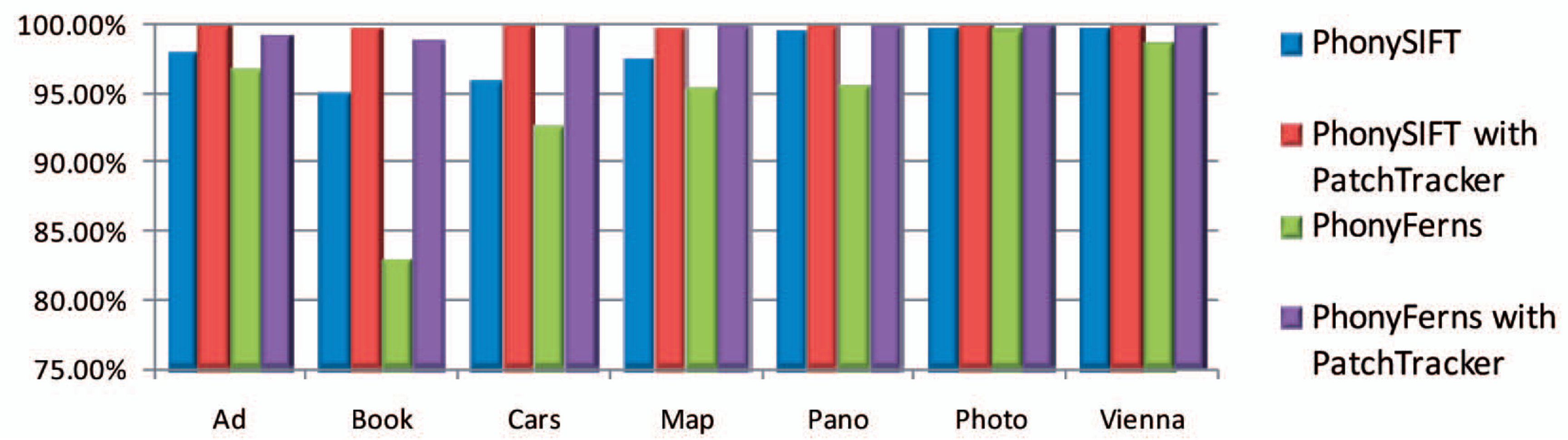
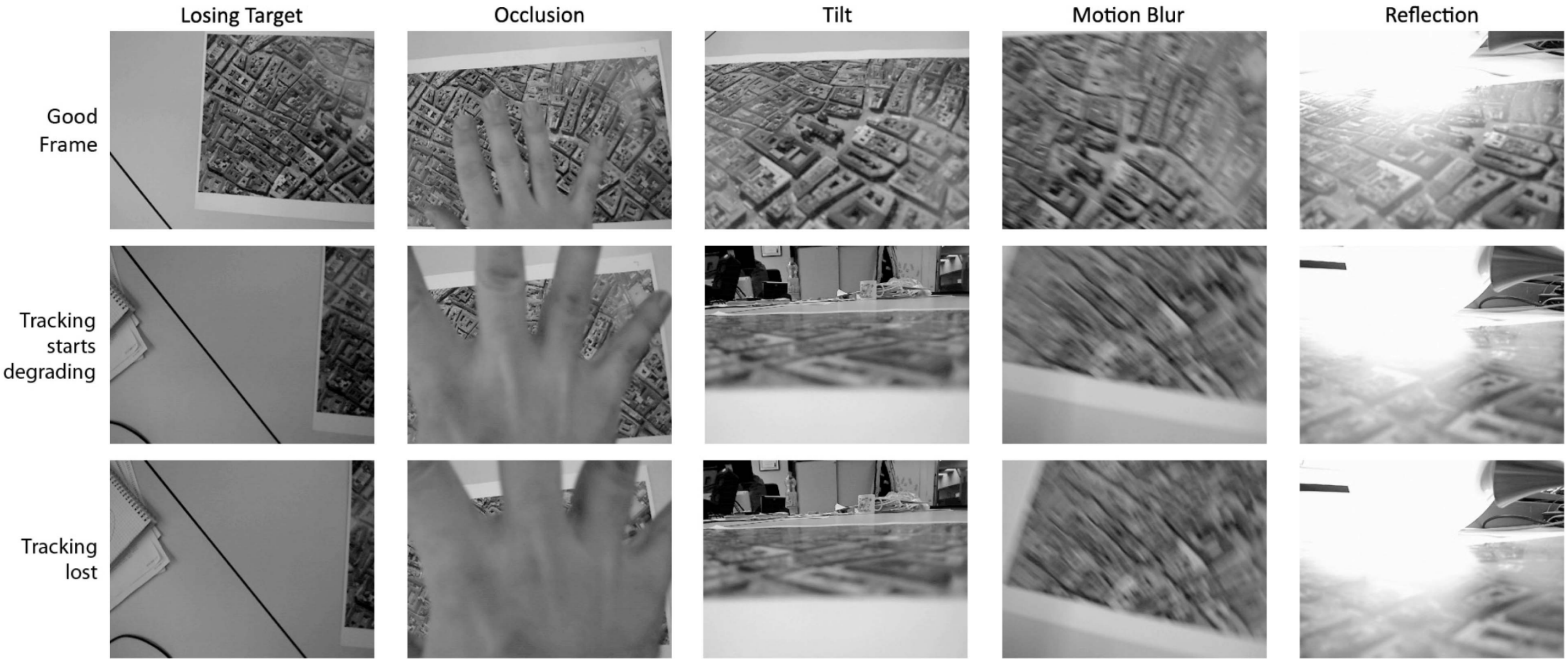


Fig. 5. The seven test sets (a)-(g): book cover, advertisement, cars movie poster, printed map, panorama picture, photo, and Vienna satellite image.

Robustness over different objects



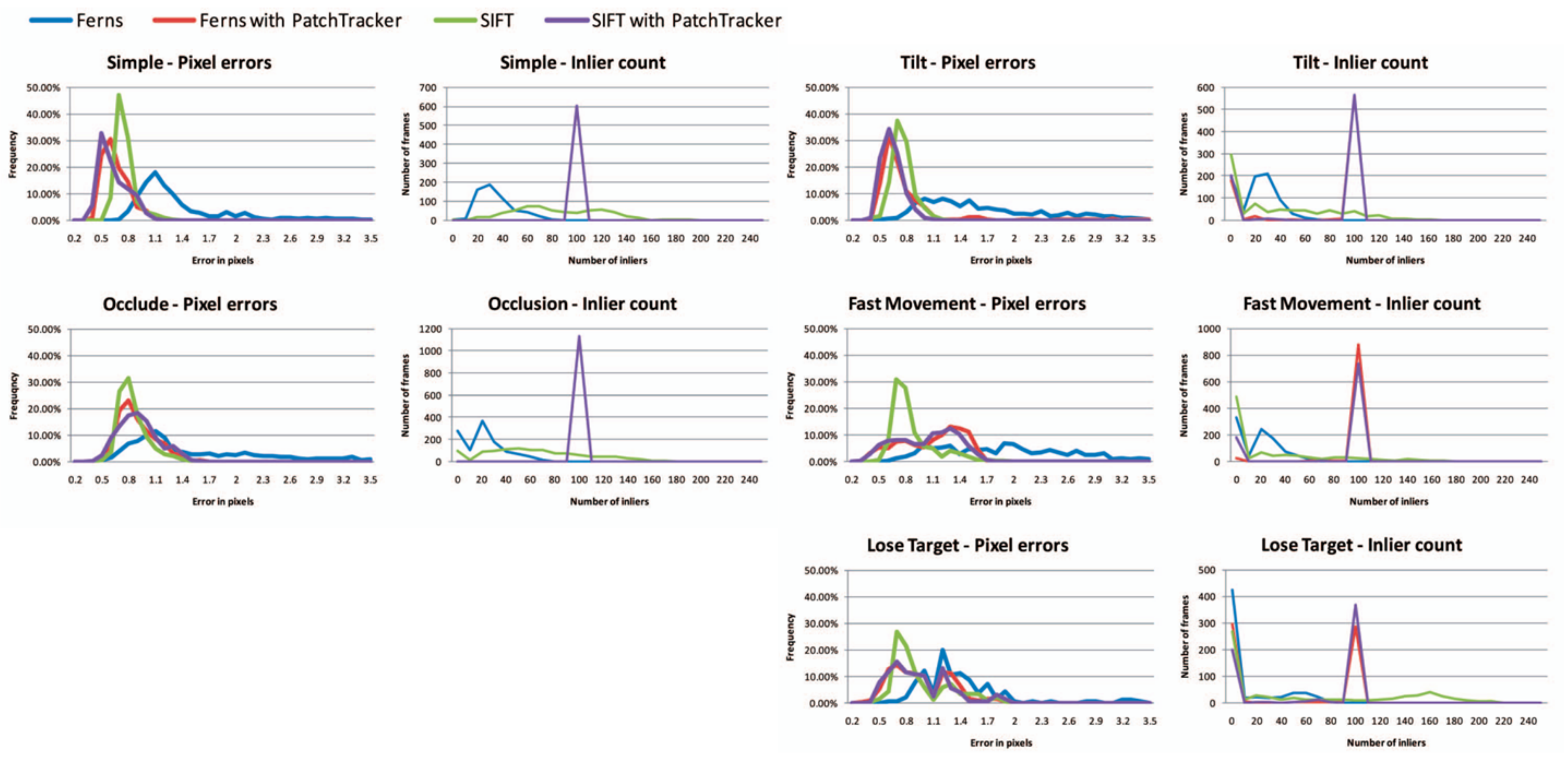
Typical Situations of Switch



Typical Situations of Switch

- (Show paper Figure 7)

Ferns vs SIFT vs PatchTracker



Detailed Speed Analysis

- PhonySIFT:
  - Corner Detection(FAST) : ~14%
  - Feature descriptor and Matching : ~74%
  - Outlier Removal : ~ 9%
  - Pose Refinement : ~ 3 %
- PhonyFerns:
  - Corner detection(FAST) : ~ 22%
  - Second Octave and Blurring : ~ 17%
  - Classification : ~ 59%
  - Outlier Removal : ~ 2%

Conclusion

- Successfully worked with tracking system on phones
- Better CPU would come out in the future. The choice of the next generation feature is unknown