# VU Augmented Reality on Mobile Devices

- Introduction – What is AR
- Interaction Techniques
- Navigation, Collaboration
- *Visualization Techniques*
- Visual Coherence
- Tracking
- Gudrun Klinker

# Ubiquitous Augmented Reality in AR-ready Environments

Prof. Gudrun Klinker Ph.D
Technische Universität München
Fachgebiet Augmented Reality (FAR)

Date: May 30th 2011
Time: 2:15pm -3:45pm

Location: TU Vienna
1040 Wien, Karlsplatz 13
Hauptgebäude HS 7, Ground Floor, Stiege VII

**Abstract**
In this talk, I will present some of our recent work as well as our vision towards providing ubiquitous Augmented Reality services in AR-ready environments. I will elaborate on the concepts of ubiquitous tracking, ubiquitous information presentation and ubiquitous manipulation leading to the vision that users will use AR as one of their means to experience and manipulate a mixed physical and virtual reality. I will report on some observations we made when we tried these concepts in real applications in which users need to keep a keen eye on primary tasks in their real environment while also using/exploring an associated virtual information space.

# Internship at Qualcomm Austria Research Center

## Offering

- Be part of a team developing mobile Augmented Reality enabling technology
- Design, implement, and verify algorithms for Augmented Reality
- Support development of new computer vision approaches for mobile devices
- 3-4 month, employment

## Skills/Experience

- Outstanding C++ and/or Java programming skills,
- expertise in object-oriented design, and
- a good foundation in one of the following areas
  - Augmented Reality & Computer Vision
  - Integration & Verification
  - Software Tools

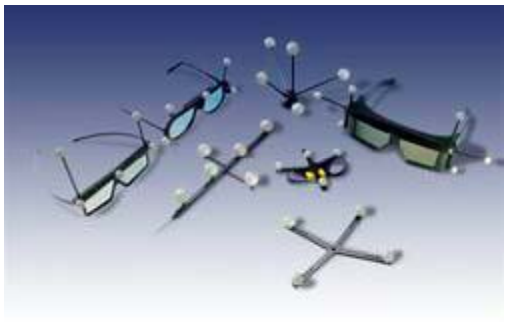
- Please apply directly on our website at www.qualcomm.com/careers/ under requisition number **E1879350.**

# Tracking

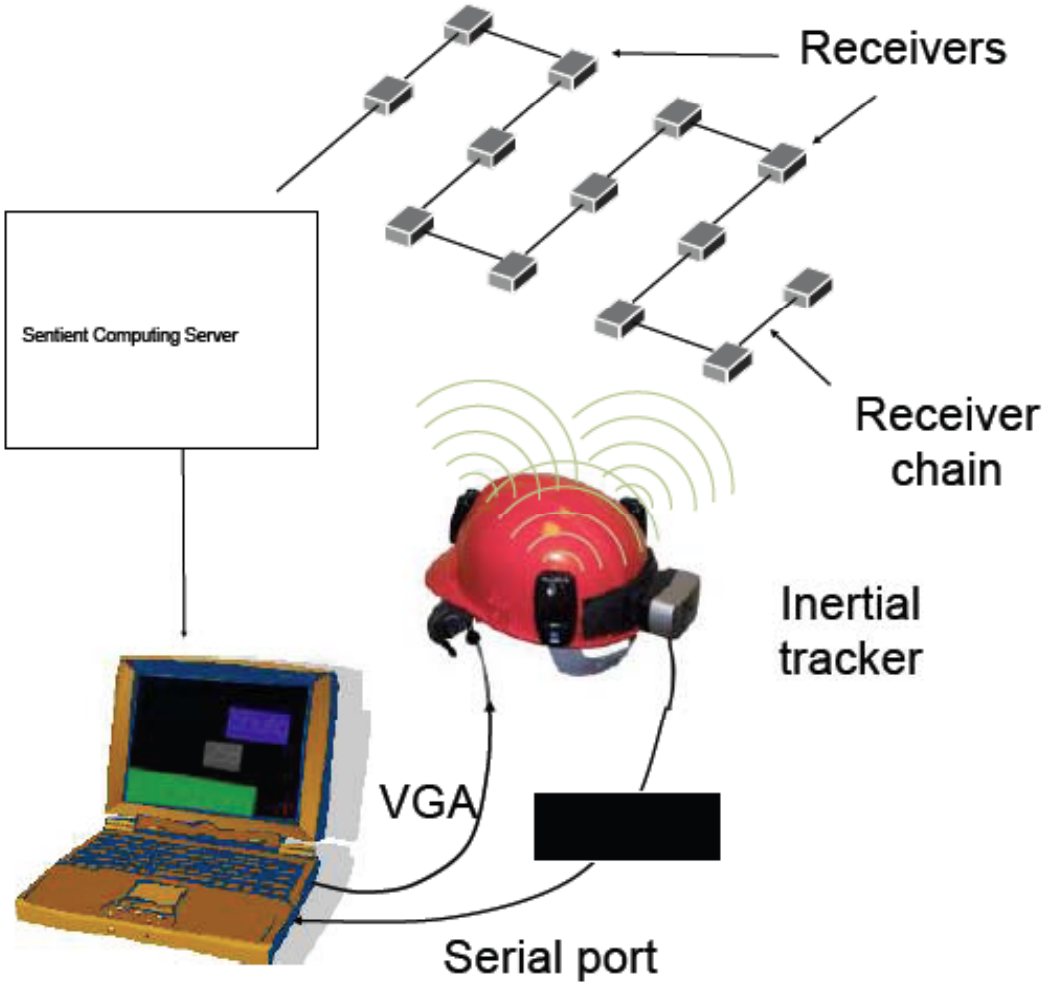
- Requirements
  - Provide position and orientation
  - Untethered
  - large working volume
  - Robustness !
  - Indoor vs. outdoor
    - Indoor: Can instrument environment
    - Outdoor: self-contained or large scale infrastructure
- No single sensor provides 6DoF
- Tracking vs. Initialization

# Indoor Tracking

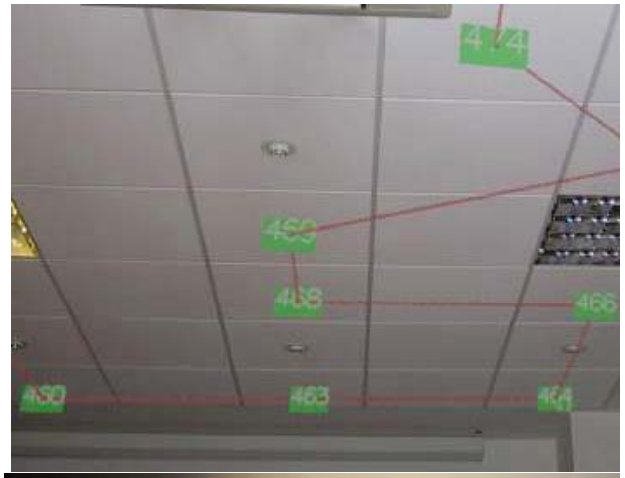
- Ultrasonic beacon array
  - AT&T Bat, Intersense IS900
- Magnetic Trackers
- Infrared LED array
  - UNC HiBall, MIT‘s locust swarm
- Outside-in computer vision
  - Observer cameras + passive IR targets (e.g., ART Track - roomsize)
- Inside-out computer vision
  - Fiducials (e.g. ARToolKit)
- Dead-reckoning techniques



# Example: AT&T Sentient AR

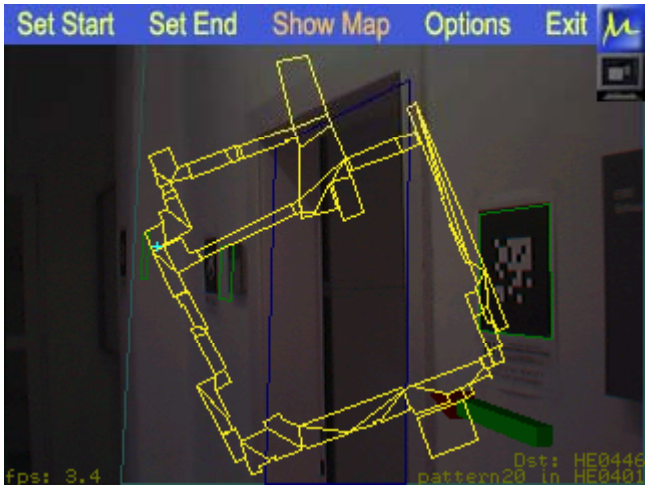






# Signpost 2003

- Guides a user trough an unfamiliar building
- Requires markers on the walls and 3D model of the building

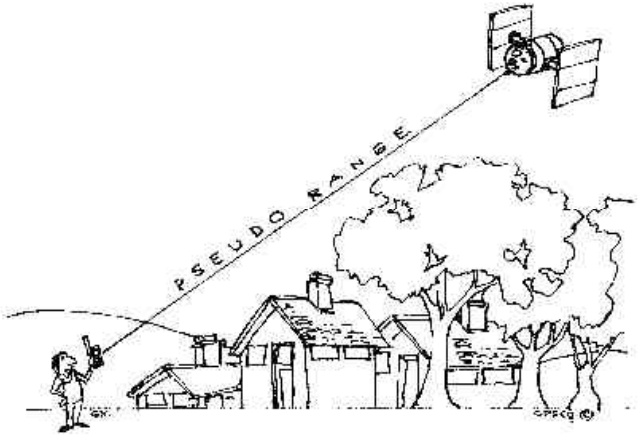



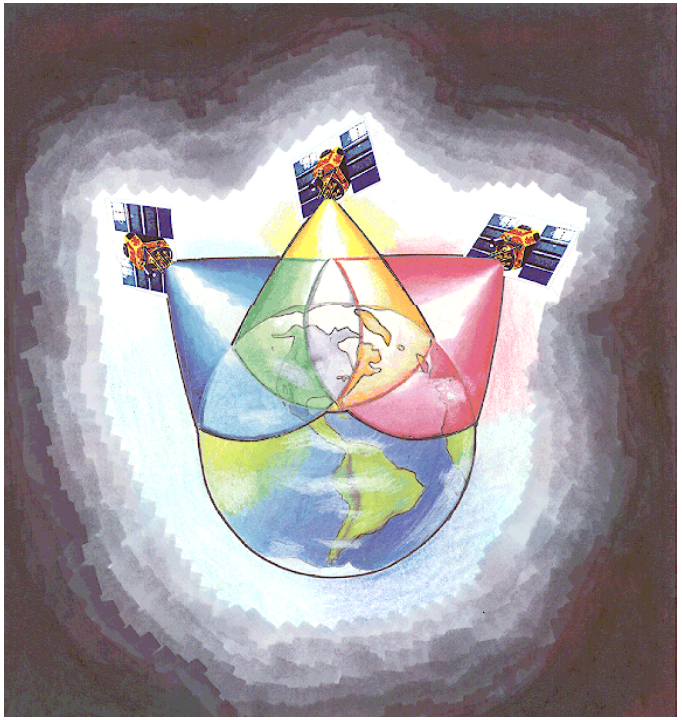
# Outdoor

- Position
  - GPS, UWB, Omnisense
  - WiFi, Cell information
- Orientation
  - Gyroscopes, magnetometers, linear accelerometers
- Inside-out computer vision
  - Natural features
  - Image databases
- Requires sensor fusion

# GPS

- Pseudo range to satellite
  - Robust, always & everywhere available
  - many error sources along the way
- Consumer level
  - 5 - 50m
  - Urban canyons, multi-path, shadowing
- DGPS
  - local correction of atmospheric errors
  - >0.5m
- RTK
  - Phase information
  - >2.5cm horizontally, >10cm height



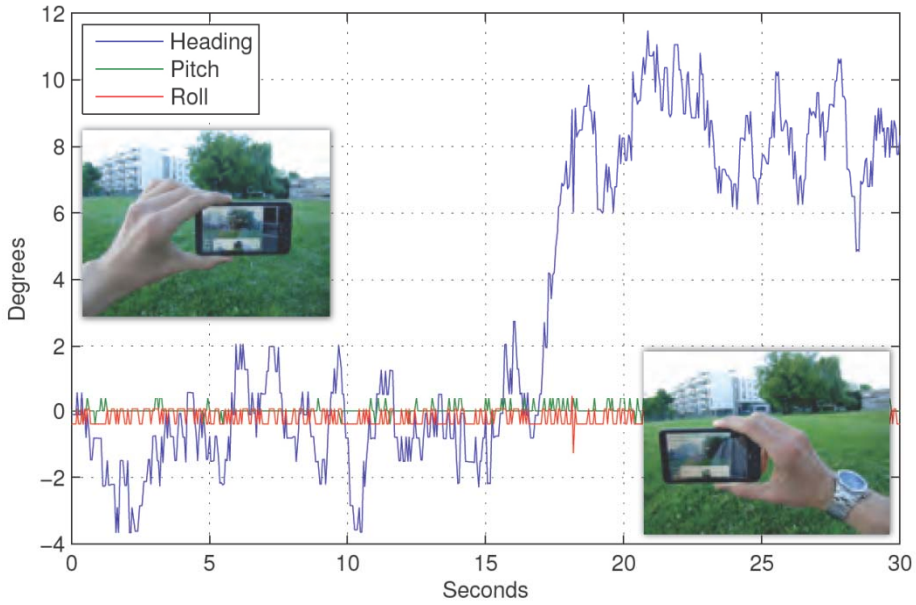



# Other RF methods

- WiFi, Cell tower
  - Skyhook, large database of WiFi and cell tower
  - Used in mobile phones
- Ubisense
  - Very short pulse signal
  - Range and direction measurements
  - 30m range, 15cm accuracy
- Omnisense
  - Coded signal
  - 4km range, 0.5m accuracy

# Orientation


- Inertial measurement units
  - Gyroscopes
  - linear accelerometers ~ gravity
  - magnetometer ~ 3D compass
- laser gyroscopes
  - Townwear, Satoh ’99
- Magnetometer ~ 3D compass
  - Deviations a common problem

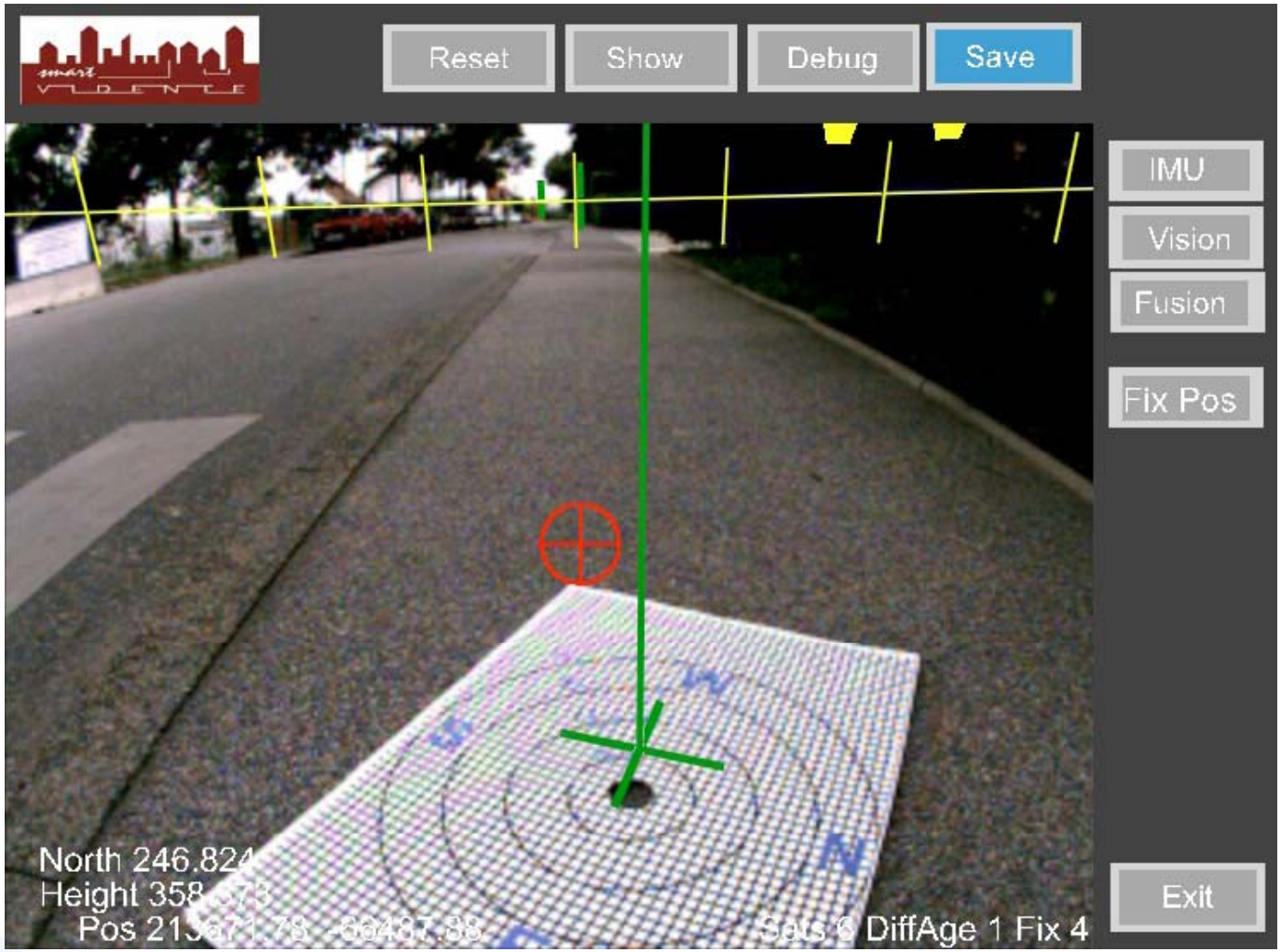


# Example: Smart Vidente

- Tablet PC J3400
  - C2Duo (SU9600 1,60GHz ,1.6 kg)
- Inertial Sensor XSENS MTx
- VRMagic VRM FC-6 COB COLOR
- RTK GPS
  - Novatel OEMV1 – L1
  - Novatel OEMV2 – L1/L2
- Differential data from network
- Kalmanfilter
- Validated to: 5cm 2D location, 10cm height

**Tablet PC based AR System**

# Reprojection error



# Handheld Information Browsers


**Wikitude**
**Geo-referenced Wikipedia information**


**Peak.ar**
**overlays names of peaks**


**Layar**
**Dedicated content layers**

# Navigation


**Wikitude Drive**
**Navigationsinformation**

**acrossair**
**Nearest Tube**

# Image-based Tracking

- Many devices have cameras !
- Pixel accuracy -> computer vision


**Marker** **Natural Features** **Image-based Localization** **Visual Search**

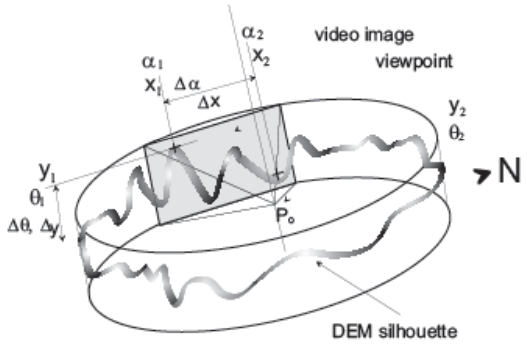
local vs. global

# Image-based Tracking

- Many devices have cameras !
- Pixel accuracy -> computer vision
- Model-based tracking
  - Natural features
  - Computer graphic models
- Requires Initialization
- Not robust
  - Sensor fusion
  - Recovery methods

# Many different methods and combinations

- Features
  - Points
  - Edges
  - Horizon
- • Sensor fusion
  - Gyroscopes




**Behringer, ’98**

**Azuma, ’99**

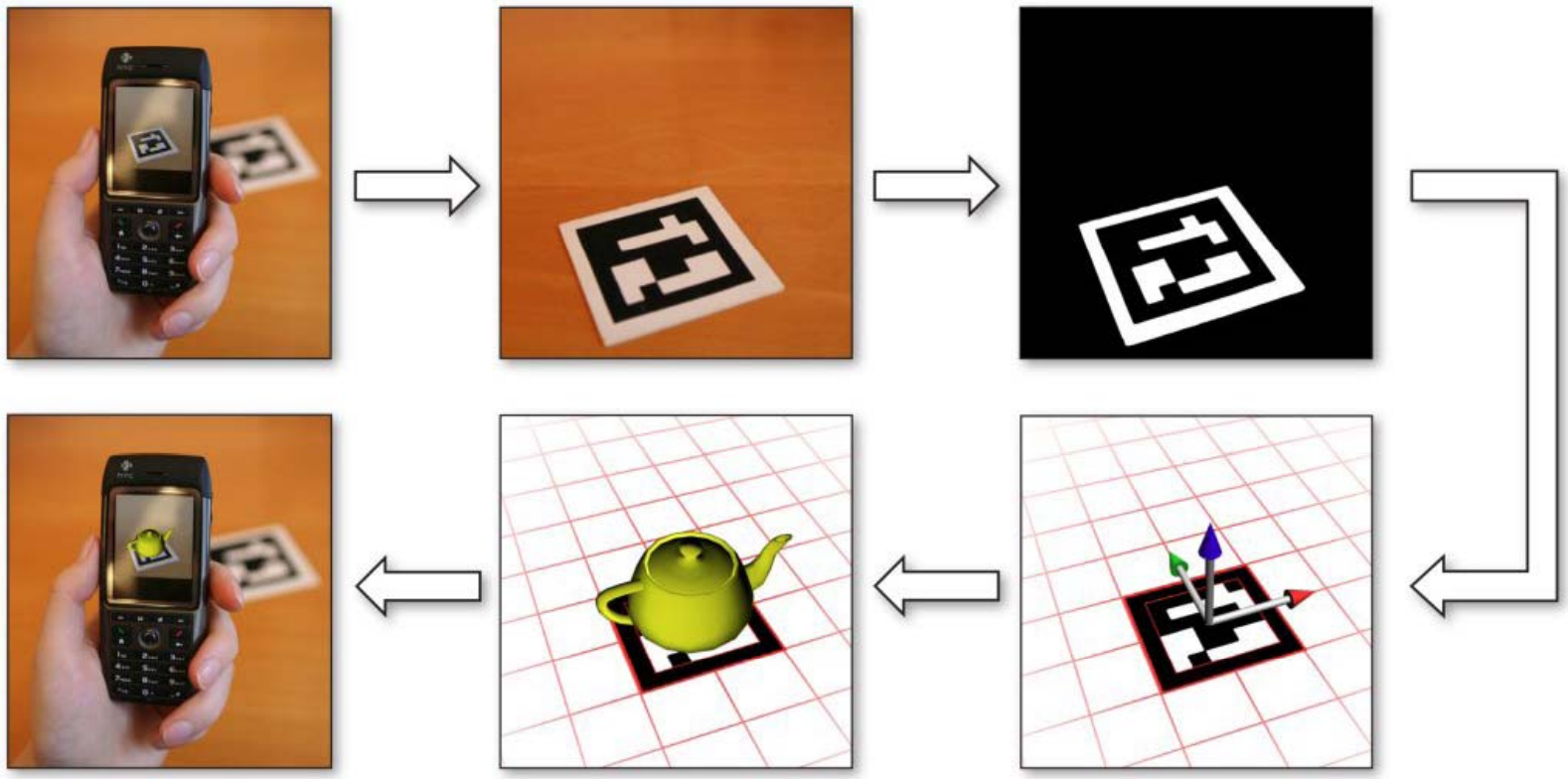
**Kretschmer et al., ’02**


**Ribo et.al., ’02**

# Starting Simple: Marker Tracking

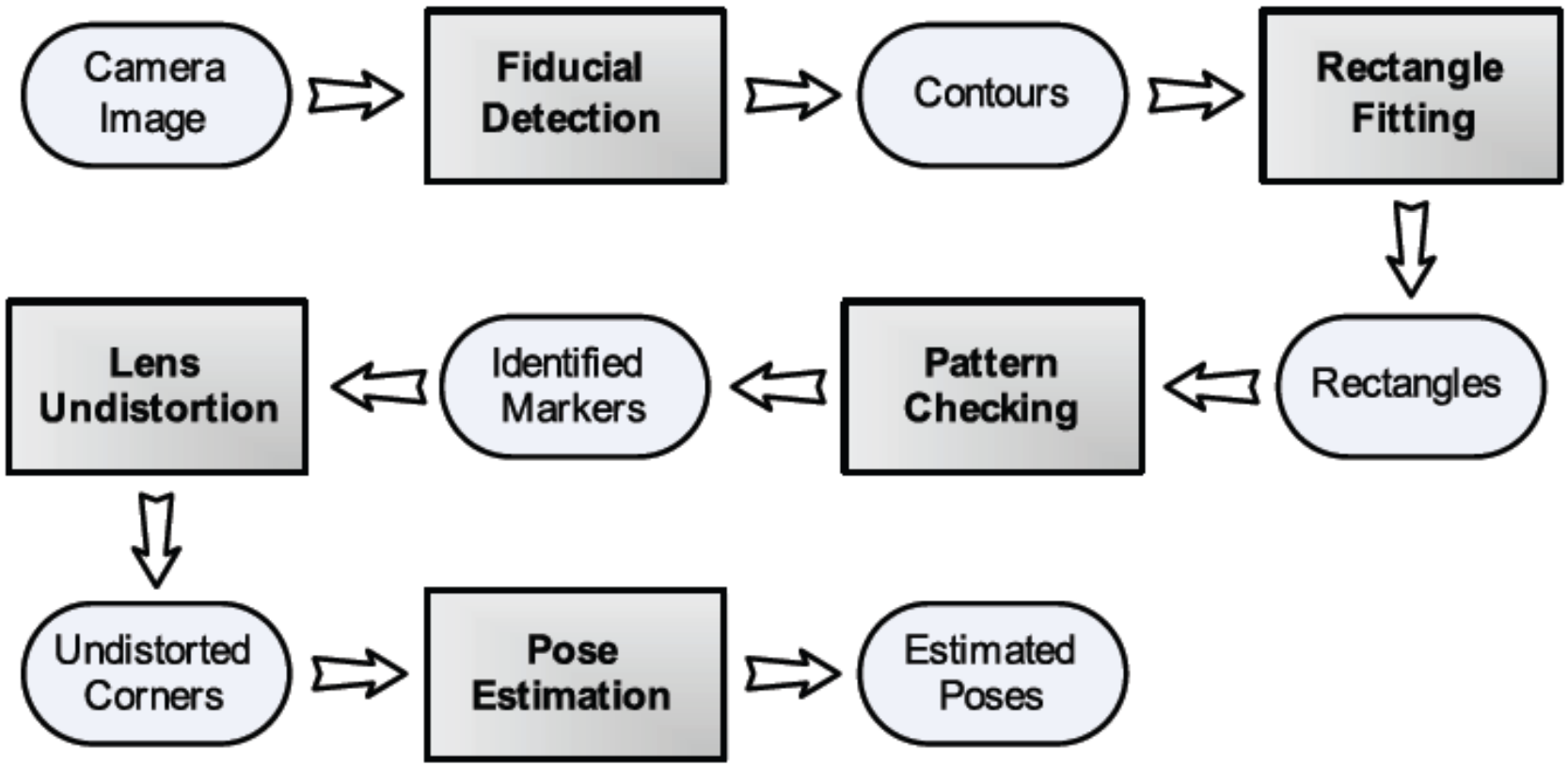
- Has been done for more than 10 years
  - Some mobile phones today are faster than computers of that time
- Several open source solutions exist
- Fairly simple to implement
  - – Standard computer vision methods
- A rectangular marker provides 4 corner points
- -> enough for pose estimation!

# Marker Tracking Pipeline Overview



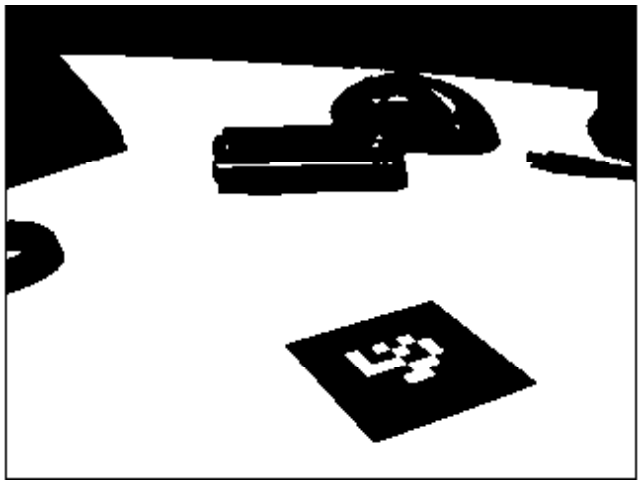
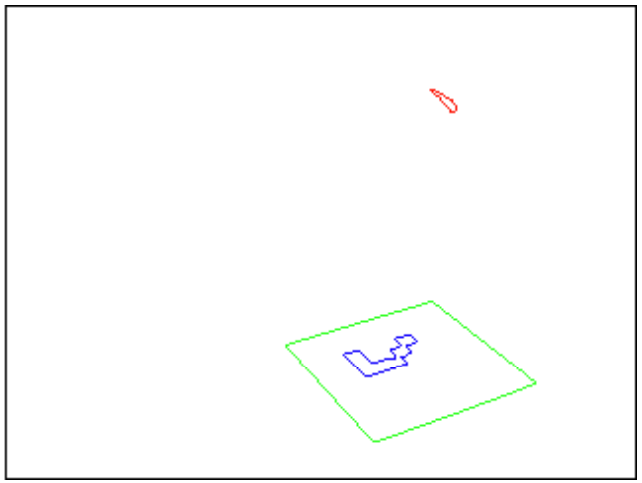
**Goal: Do all this in less than 20 milliseconds on a mobile phone…**

# Marker Tracking – Overview



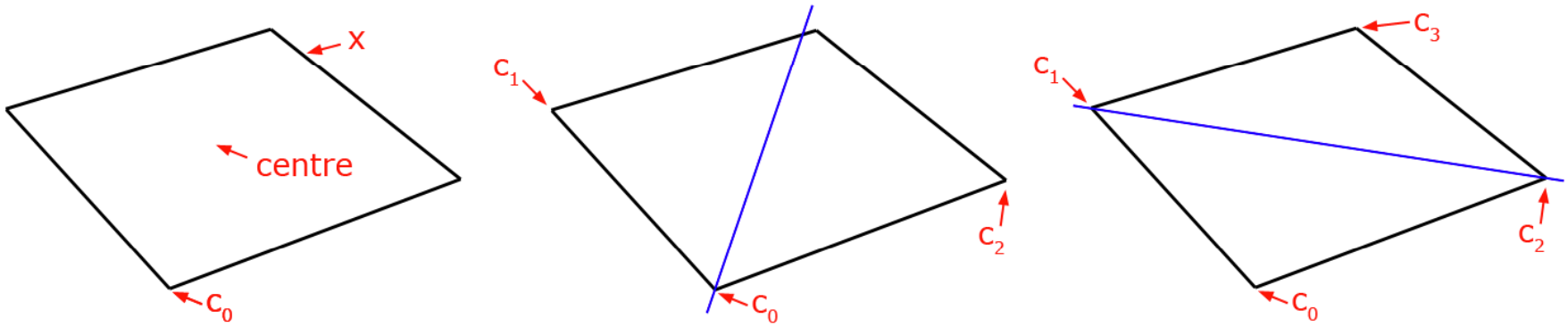
# Marker Tracking – Fiducial Detection

- Threshold the whole image
- Search scan-line per scan-line for edges (white to black steps)
- Follow edge until either
  - Back to starting pixel
  - Image border
- Check for size
  - Reject fiducials early that are too small

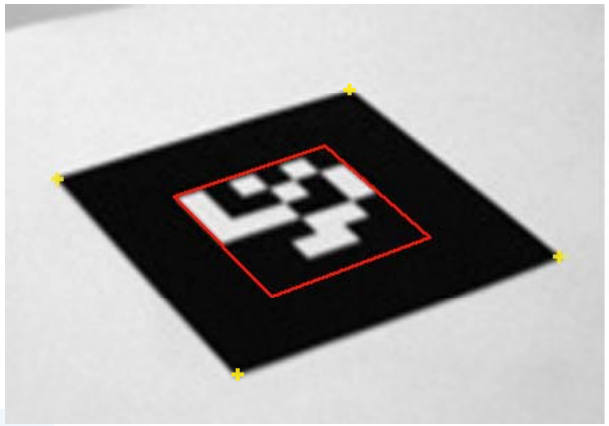
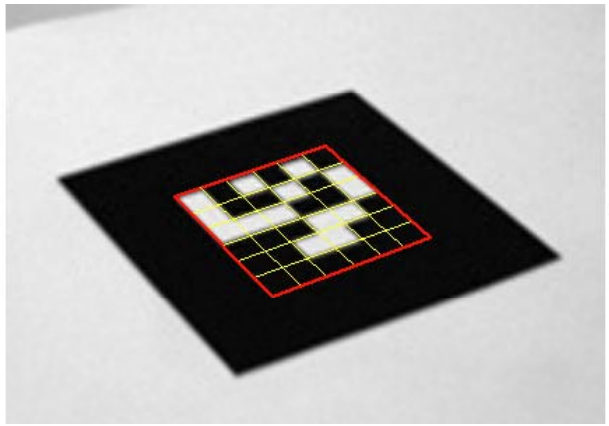
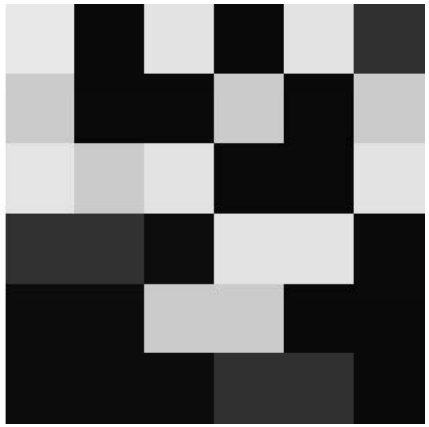
# Marker Tracking – Rectangle Fitting

- Start with an arbitrary point “x”
- The point with maximum distance must be a corner c0
- Create a diagonal through the center
- Find points c1 & c2 with maximum distance left and right of diagonal
- New diagonal from c1 to c2
- Find point c3 right of diagonal with maximum distance
- Repeat to check if no more corners exist



# Marker Tracking – Pattern checking

- Calculate homography using the 4 corner points
  - “Direct Linear Transform” algorithm
  - Maps normalized coordinates to marker coordinates (simple perspective projection, no camera model)
- Extract pattern by sampling
- Check pattern
  - Id (implicit encoding)
  - Template (normalized cross correlation)
- Four 2D-3D correspondences ~ pose estimation

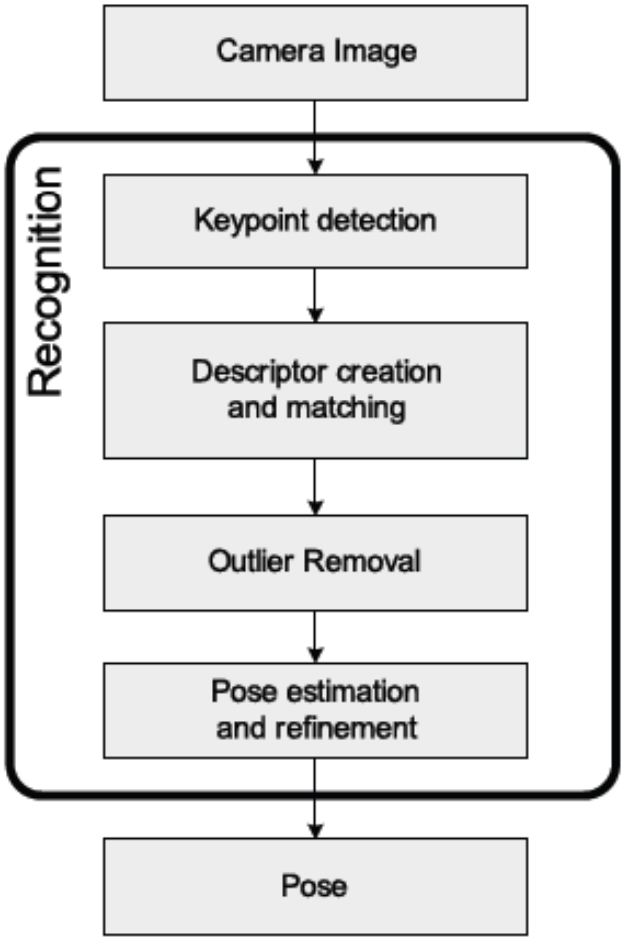




# Natural Feature Tracking

- More difficult than marker tracking
  - Markers are designed for their purpose
  - The natural environment is not…
- • Less well established methods
  - Every year new ideas are proposed
- • Usually much slower than marker tracking

# Detection in every Frame

- This is what most „trackers“ do…
- Targets are detected every frame
- Popular because detection and pose estimation 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
  - Memory intensive (requires ~10x too much memory for phones)
  - Long training phase

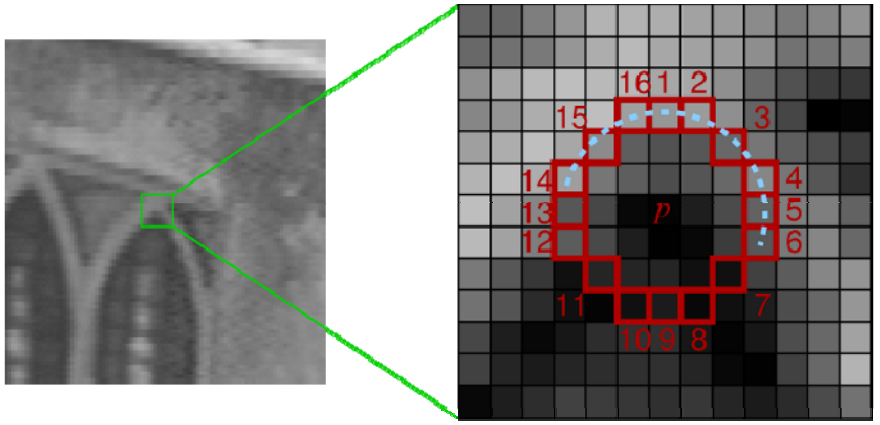
**SIFT: [Lowe, 2004]** **Ferns: [Ozuysal, 2007]** **See [Wagner, 2008]**

# Overview on SIFT descriptor

- key-point localization
  - Searching for minima and maxima in scale space (also provides scale estimation for scale invariance)
  - Requires building pyramid of Difference of Gaussian
  - Major performance bottleneck of SIFT
- Feature description
  - Estimation of dominant 2D feature orientations
  - Orientation histogram of 4x4 sub-regions (128 bins)
- Feature matching
  - Feature database stored as k-d tree for sub-linear search time

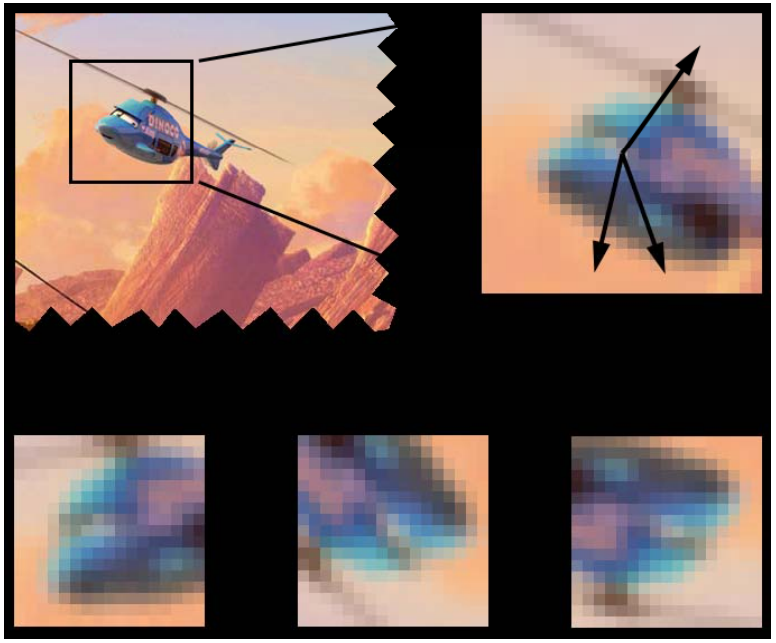
# PhonySIFT – key-point Detection

- Scale-space search way too slow…
- Replaced with FAST* corner detector
- Robustness by database on multiple scales

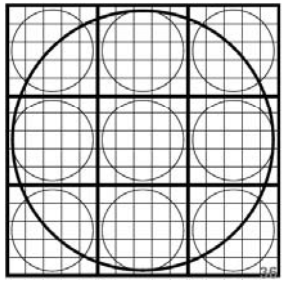


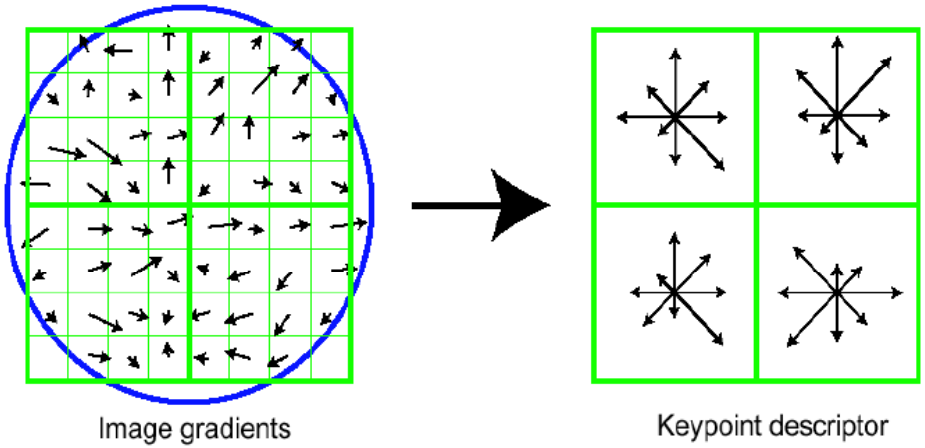

*** [Rosten, 2005]**

# PhonySIFT – Descriptor Creation



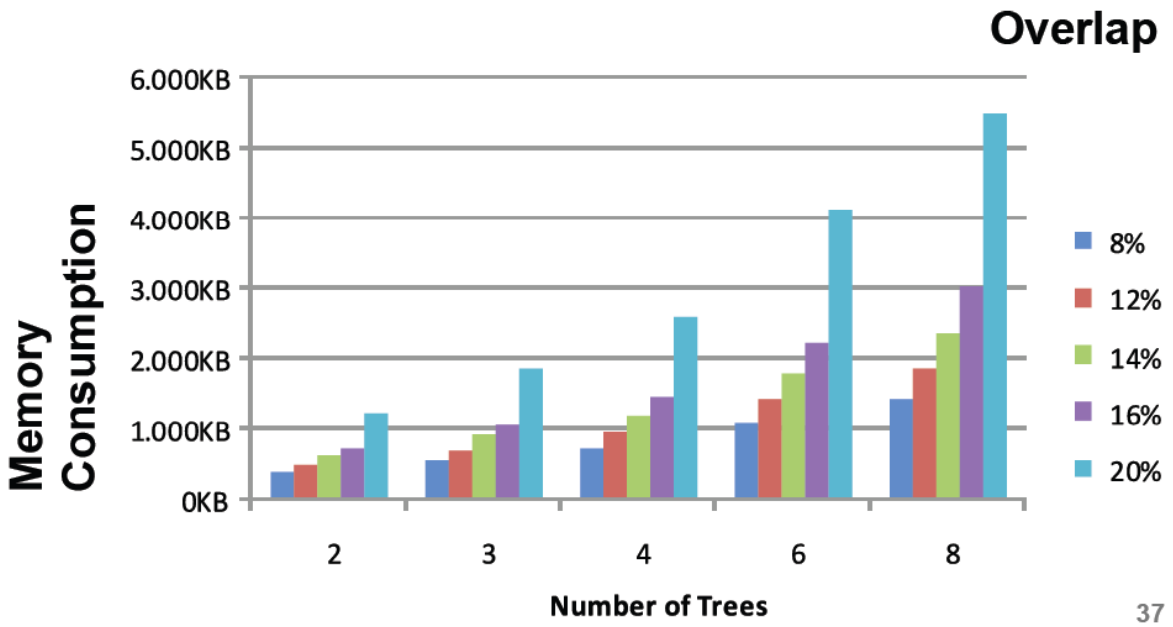
- Estimate dominant key-point orientation using gradient histogram
- Compensate for detected orientation
- Create descriptor from 3x3 subregions with 4 directions = 36 bins instead of 128
  - – All weights pre-computed!





# PhonySIFT – Descriptor Matching

- Brute force matching not an option
- K-d tree still not efficient enough
- New approach: Spill forest
  - Multiple spill trees (k-d tree with overlap)
  - Randomized dimensions for pivoting
  - Voting to merge results from different trees

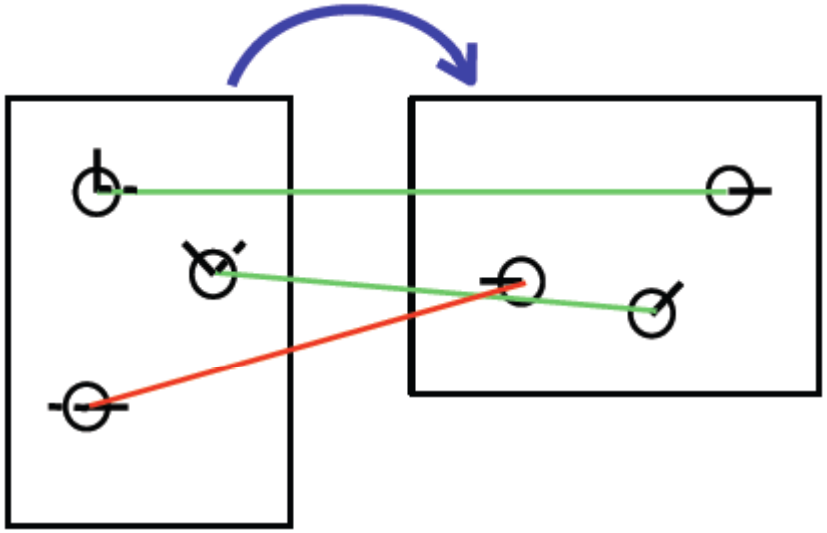


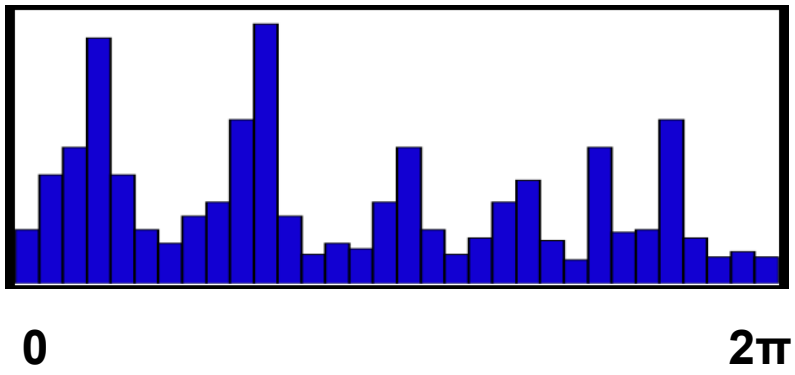
# PhonySIFT – Outlier Removal

- Cascade of removal techniques
- • Start with cheapest, finish with most expensive…
  - – Overall rotation check
  - – Line tests
  - – RANSAC for homography estimation

# Overall rotation check

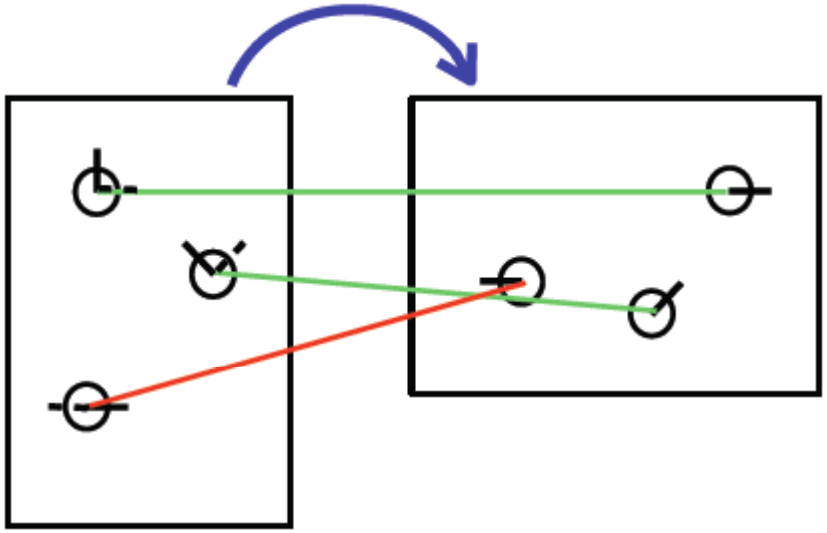
- SIFT provides key-point rotation for free
  - – All key-points must have same relative rotation
  - – Look at histogram and keep only peaks

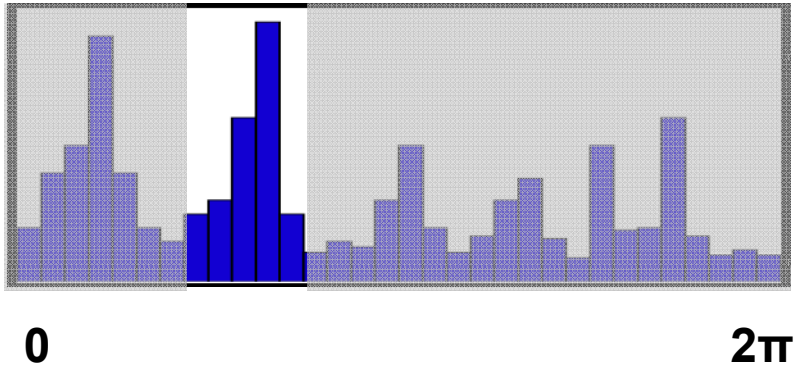




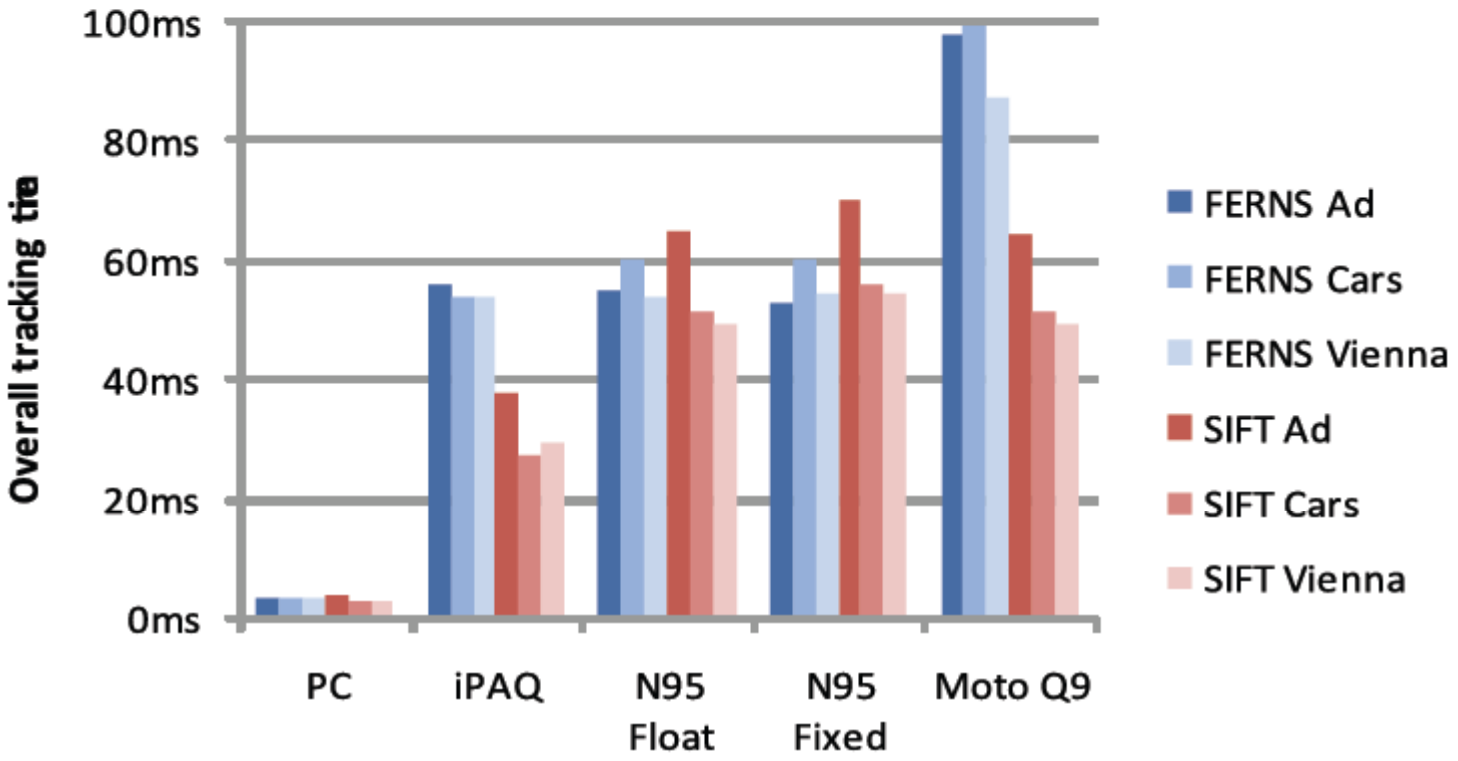
# Overall rotation check

- SIFT provides key-point rotation for free
  - – All key-points must have same relative rotation
  - – Look at histogram and keep only peaks

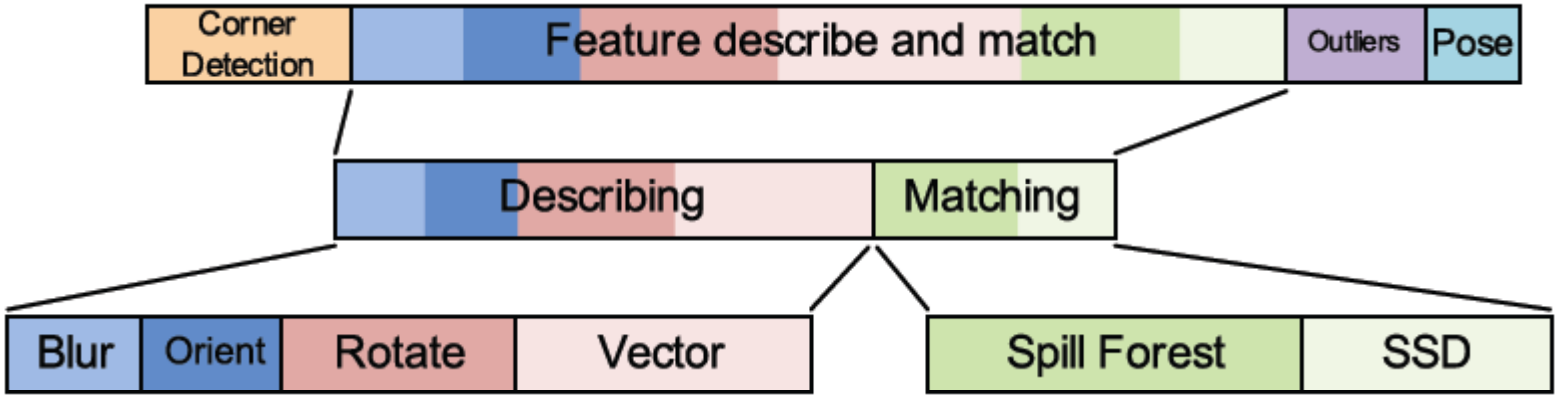




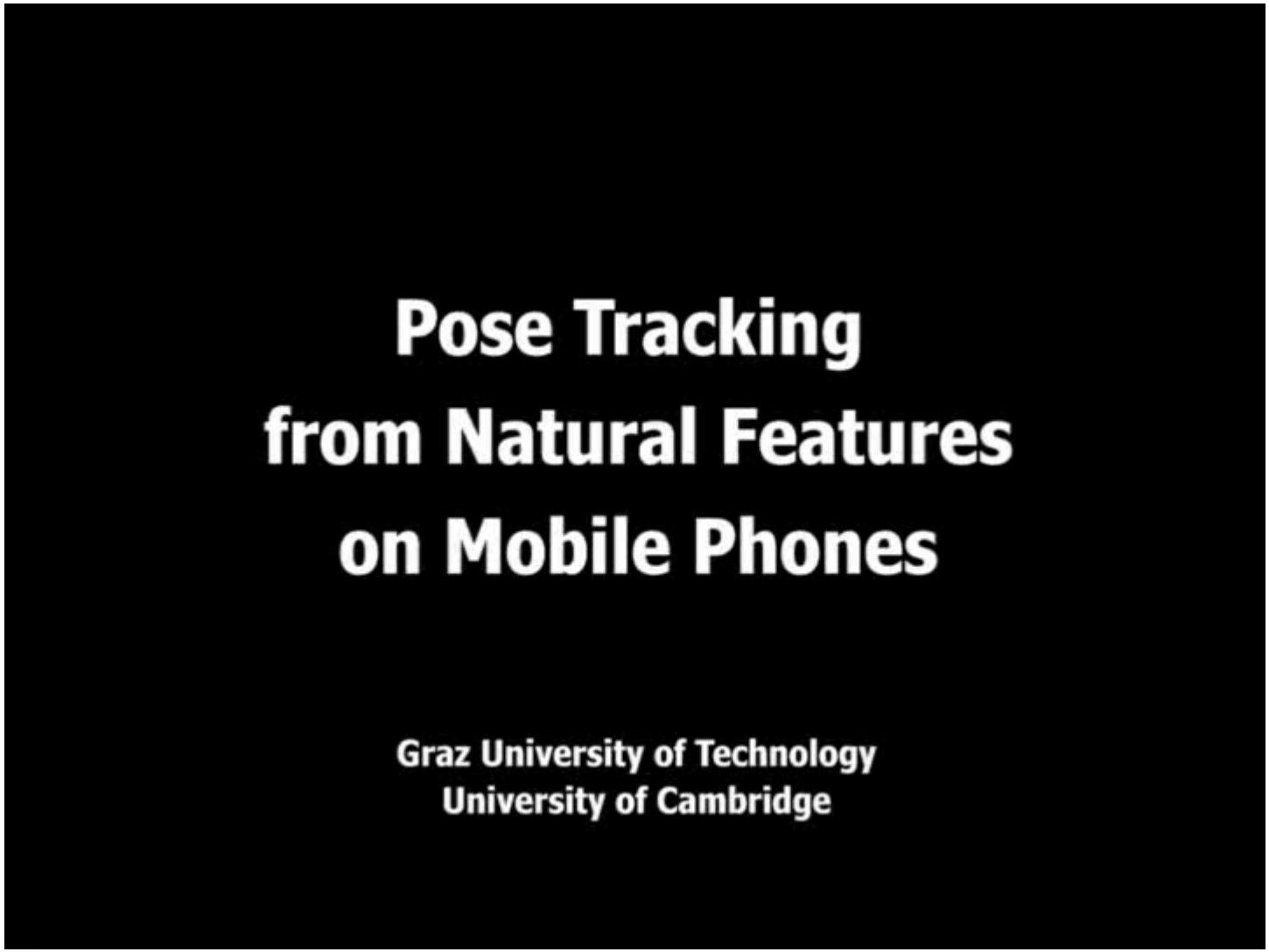
# Performance of SIFT and Ferns modified for mobile phone tracking



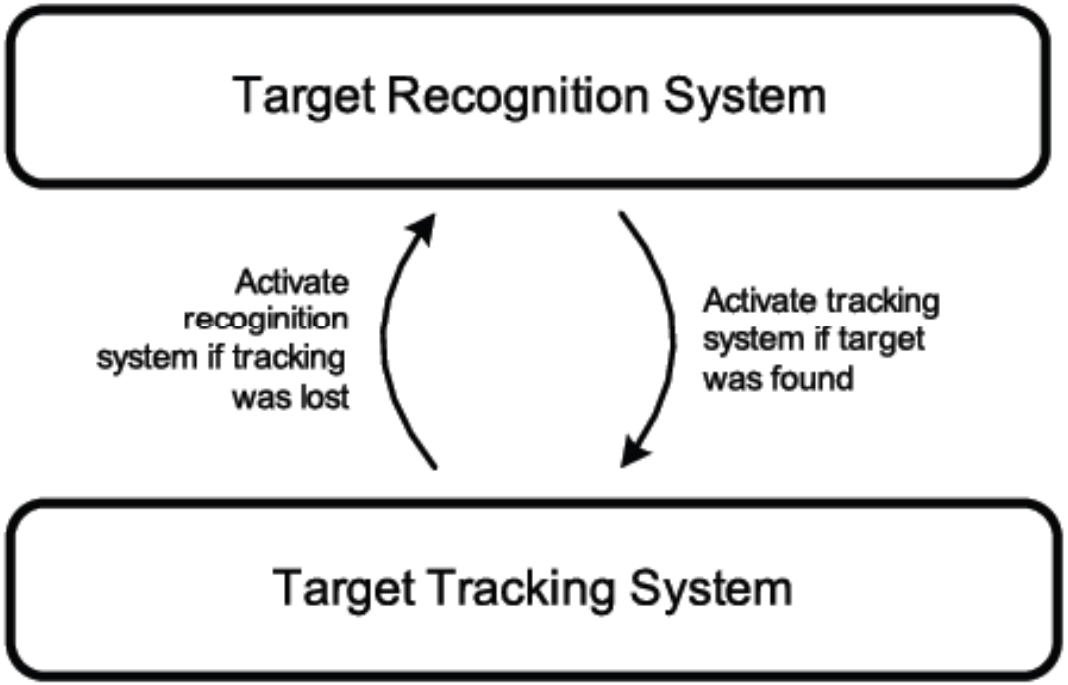
# Speed Analysis for PhoneSIFT



# NFT with SIFT on a Mobile Phone



# Doing it better:
# Dedicated Detection and Tracking

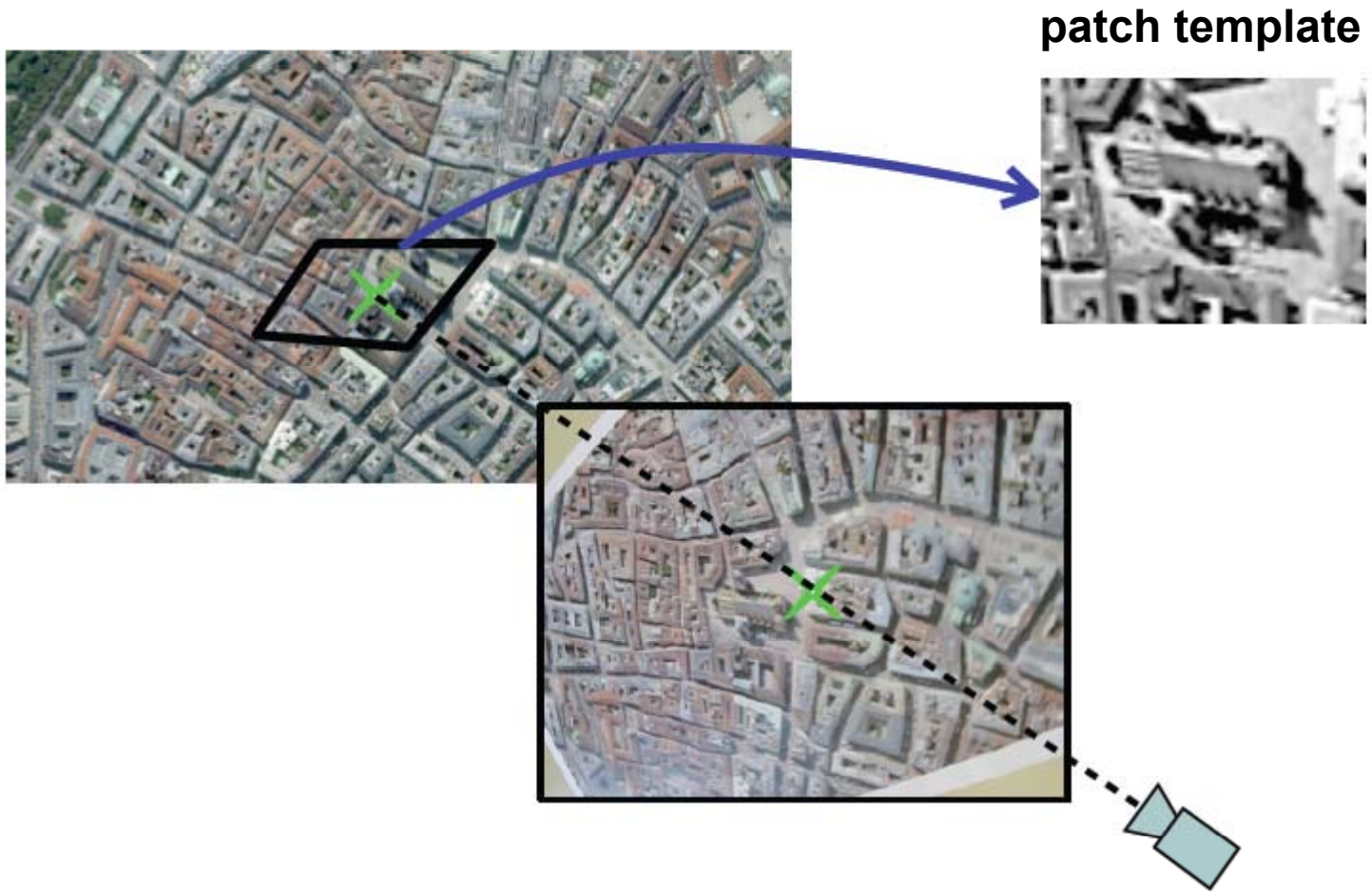


# PatchTracker Workflow (1)

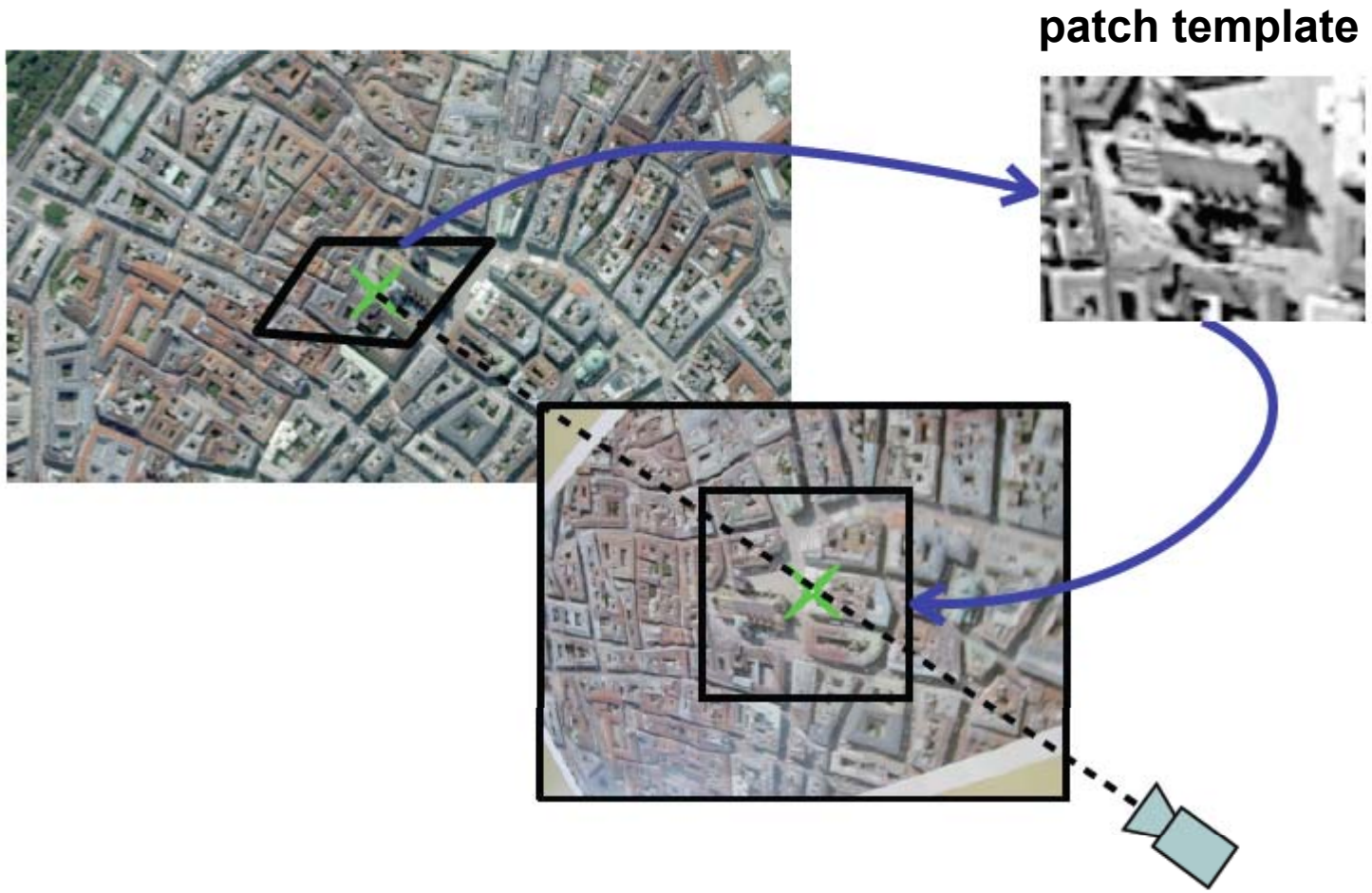
- Target Recognition:
  - find feature in reference image
- Target Tracking:
  - Take previous pose and apply motion model
- Get estimate for what we are looking for
  - Create affine warped patches of reference features
- Closely resemble how the feature should look in the camera image
  - Project patches into camera image and use normalized cross correlation (NCC) to match

# Patch Prediction and Warping

# Patch Prediction and Warping

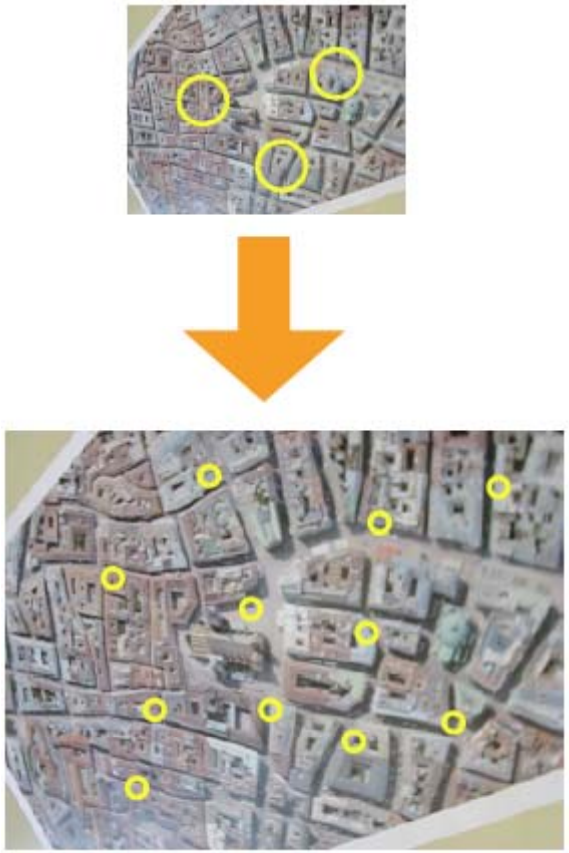


# Patch Prediction and Warping



# Patch motion measurements

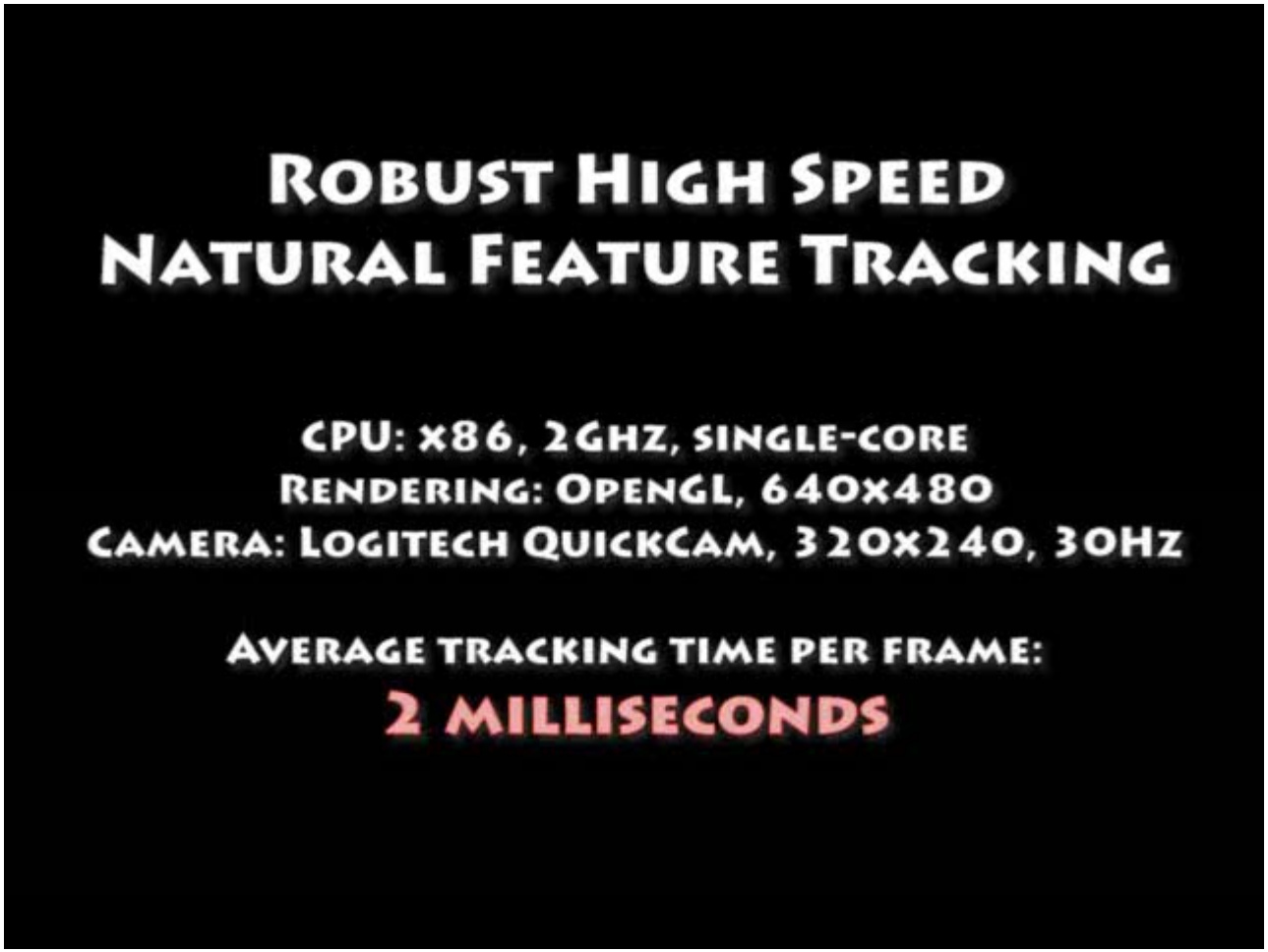
- Search half-res with low number of feature
  - E.g. 25 features, search radius of 5 pixels
- Refine estimated pose
- Search full-res with many features
  - E.g. 100 features, search radius of 2 pixels
- Refine for final pose



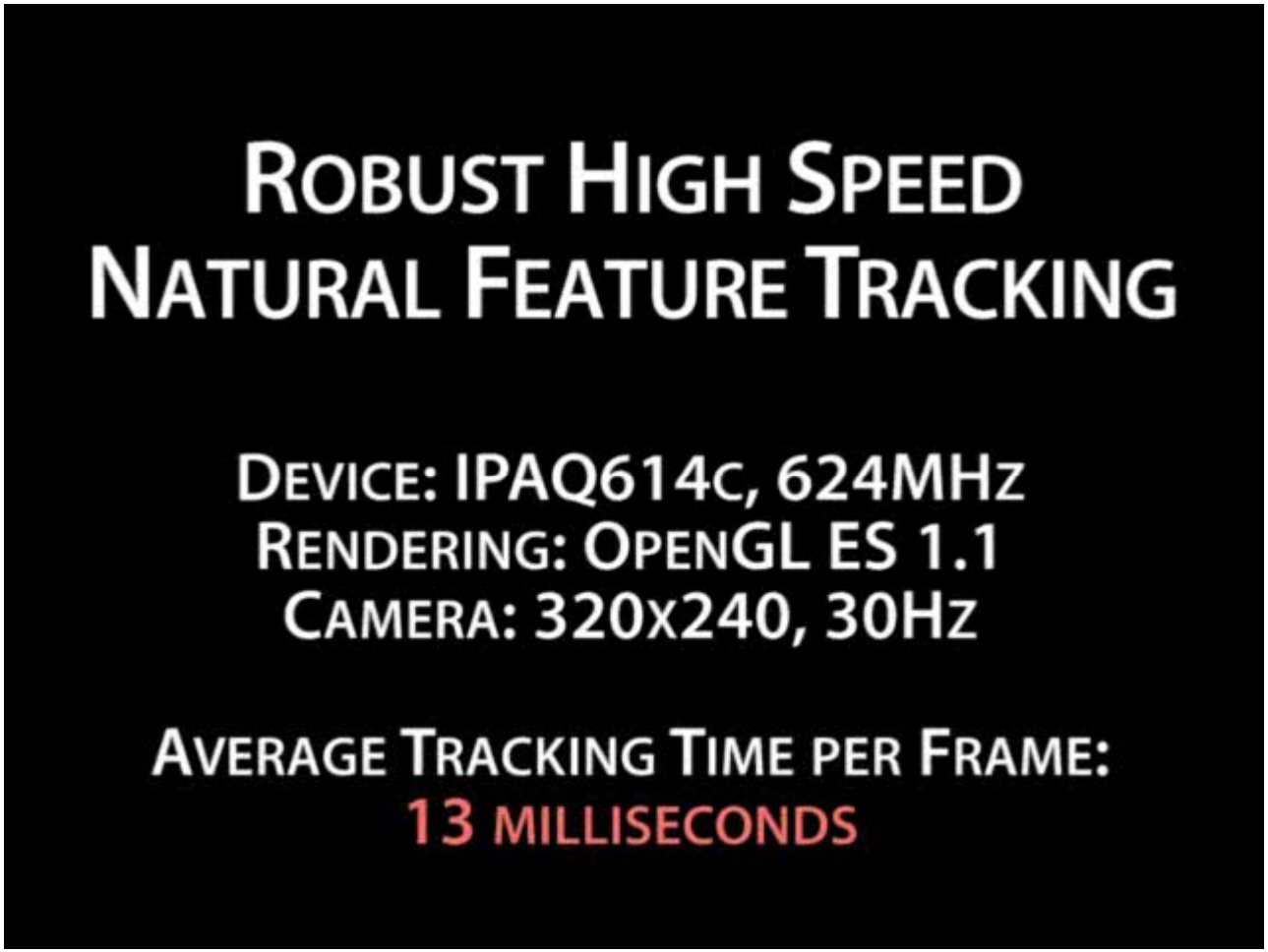
# PatchTracker Workflow Analysis

- Affine warped patches allow very strong affine transformations (tilt close to 90°)
- NCC allows severe lighting changes
- 5 pixel search radius at half-res allows “wide” base-line
  - 5 x 2 x 20Hz = 200 pixels/sec
- Tracking 100 features at full-res strongly reduces jitter

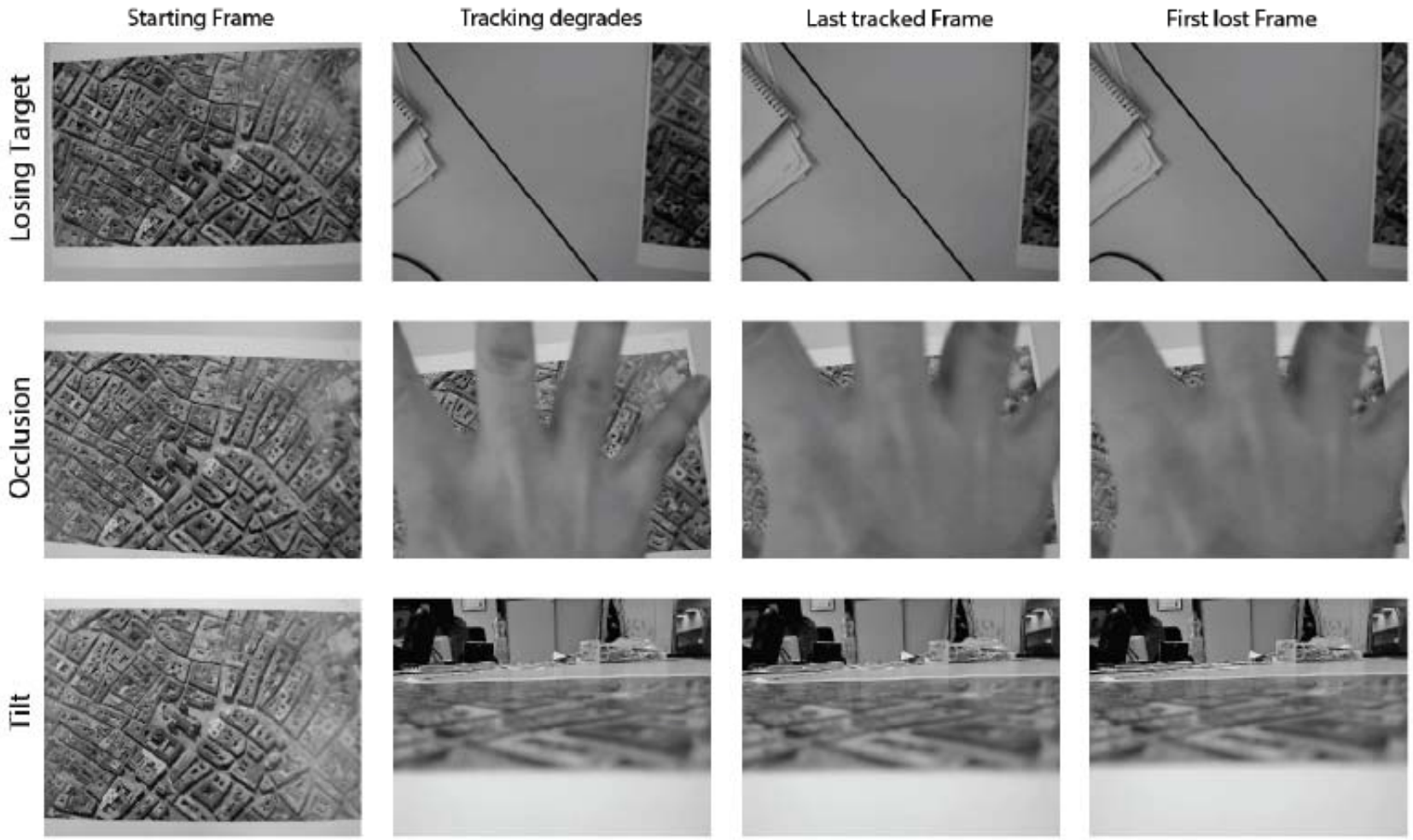
# Detection and Tracking



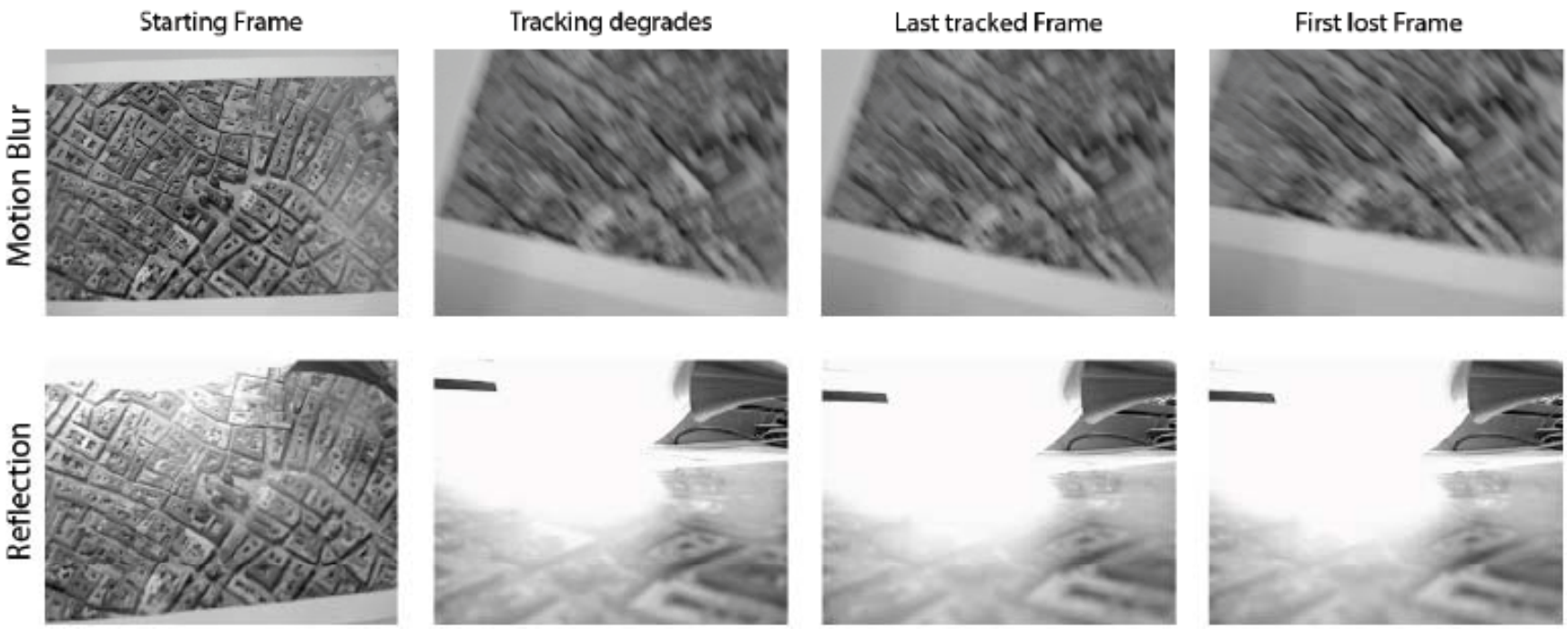
# PatchTracker in Action (2)



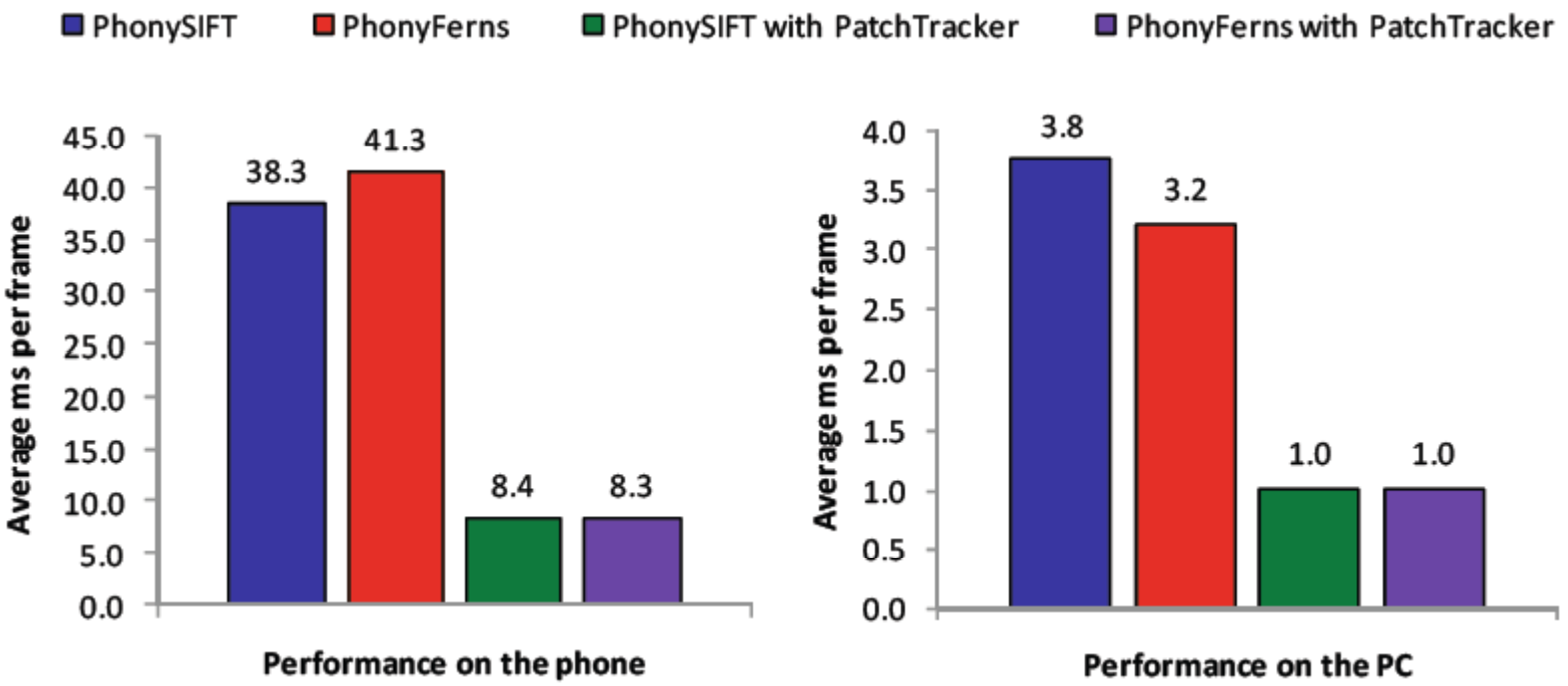
# When does it break?



# When does it break?



# How fast is it?



# Orthogonal Strengths and Weaknesses

| | SIFT/Ferns | PatchTracker |
|---|---|---|
| Recognize many targets | ☑ | ☒ |
| Detect target | ☑ | ☒ |
| Initialize tracking | ☑ | ☒ |
| Speed | ☒ | ☑ |
| Robust to blur | ☒ | ☑ |
| Robust to tilt | ☒ | ☑ |
| Robust to lighting changes | ☐ | ☑ |

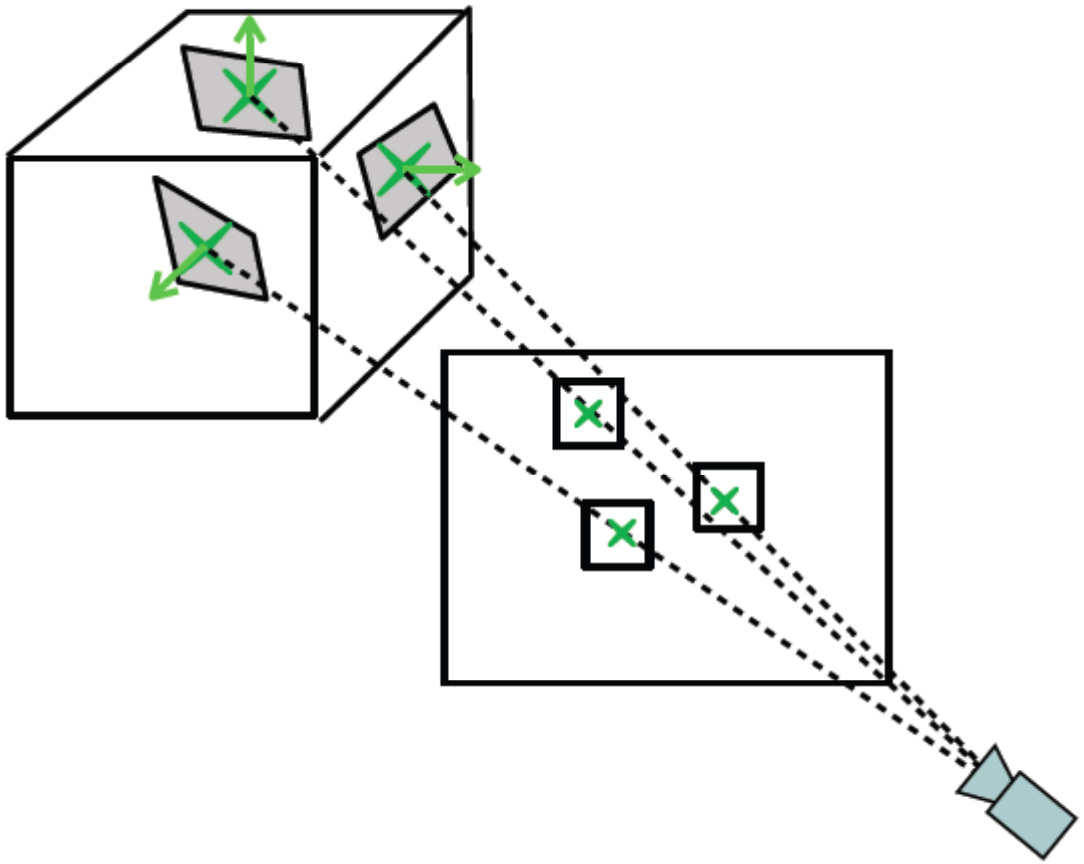
# High Robustness at fixed camera frame rate

# No need to stay in 2D

- Patches don’t need to come from a single plane
- Per feature
  - 3D position
  - Plane normal for affine warp
  - Multiple reference images
- Continue as before ...

# Features in 3D

- Different geometry per feature
- Defined by 3D shape



# Tracking 3D Objects