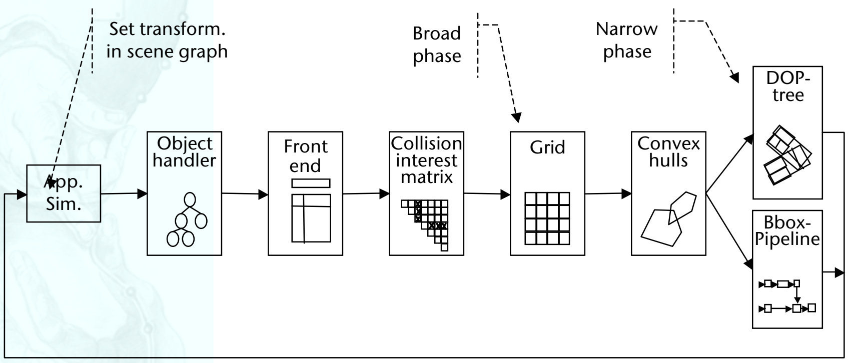
Collision Detection Towards Natural Interaction in VEs

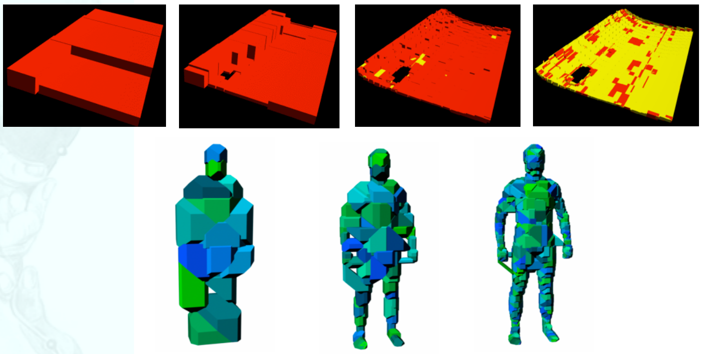
Prof. Gabriel Zachmann
Clausthal University, Germany
zach@in.tu-clausthal.de

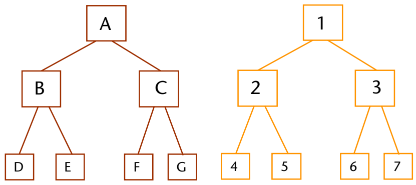
Volkswagen Tagung, Sep 2006

Collision Detection Pipeline



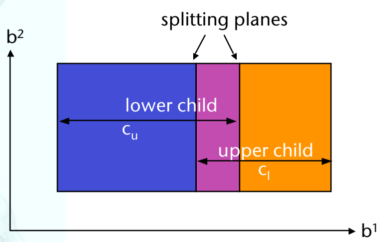
Hierarchical Collision Detection

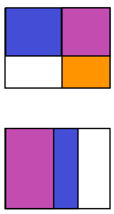




Restricted Boxtrees

- Observation: on most sides, child boxes almost touch parent box
- Combination of k-d tree and AABB

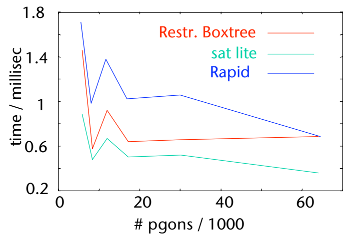


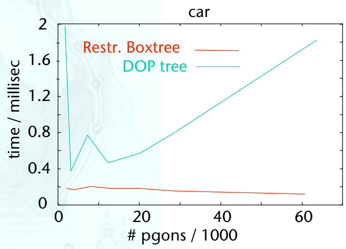


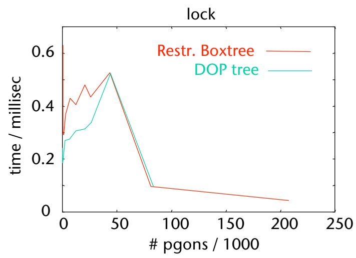
- Storage: 1 float, 1 axis ID, 1 pointer

Results

BVH	Bytes / Node
Restricted Boxtree	9
Sphere Tree	20
AABB Tree	28
OBB Tree	52
24-DOP-Tree	100



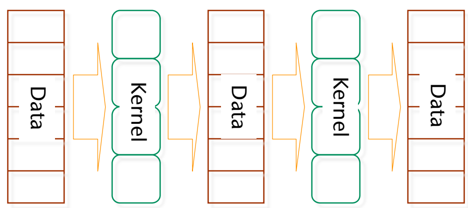




Object-Space Coll. Detection on the GPU

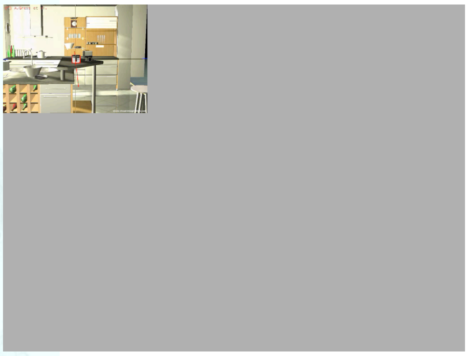
- Background on stream architectures (and GPUs)
  - Stream Programming Model =
    "Streams of data passing through computation kernels."
  - *Stream* = ordered set of data of arbitrary datatype.
  - *Kernel* = program to be performed on *each* element of the input stream
- Sample stream program:

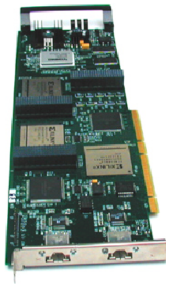
```
{
    stream A, B, C;
    ...
    kernelfunc1( input: A,
                 output: B);
    kernelfunc2( input: B,
                 output: C);
   ...
}
```



Overview

- Simultaneous overlap testing of multiple BVs
- Implementation:
  - Stream = list of BVs = texture
  - Kernel = BV intersection test = fragment program

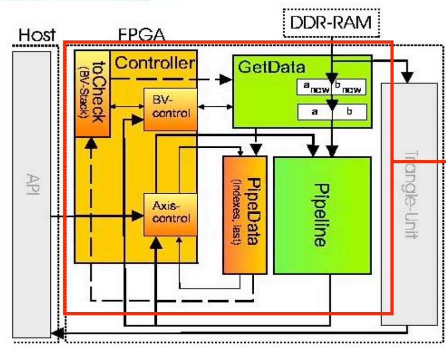


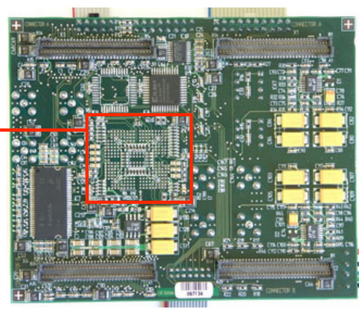


Dedicated Hardware for Coll.Det.

universitätbonn

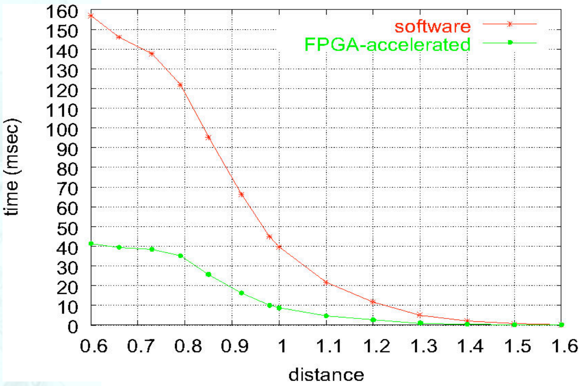
- General problem of "general purpose" computations on the GPU — competition among resources





- FPGA board (Xilinx Virtex II Pro) for prototyping

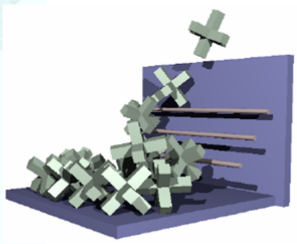
Results



- FPGA implementation has no cache yet(!)
- FPGA is much slower than ASIC (100 MHz, slow mem interface)
- With FPGA, the CPU is completely idle

Further Speedup

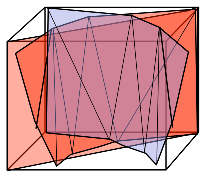
- Observation: absolute accuracy is often not necessary



→ New notion: approximative collision detection

- Goal: continuous and controlled balancing between running time and accuracy
- Benefit: time-critical computation

"Gedankenexperiment"



General Approach

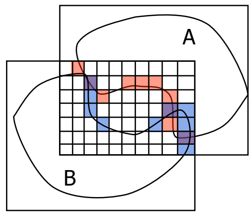
- Average-Case approach:
  - Estimate probability of intersection for whole sets of polygons (at inner nodes of BVH)

→ BVH traversal guided by probability (P-Queue)

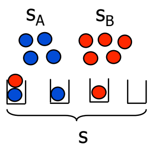
- Modification of BVHs: store simple description
- Advantage of our approach: can be applied to (almost) any kind of BVH / hierarchical collision detection

Estimating the Probability

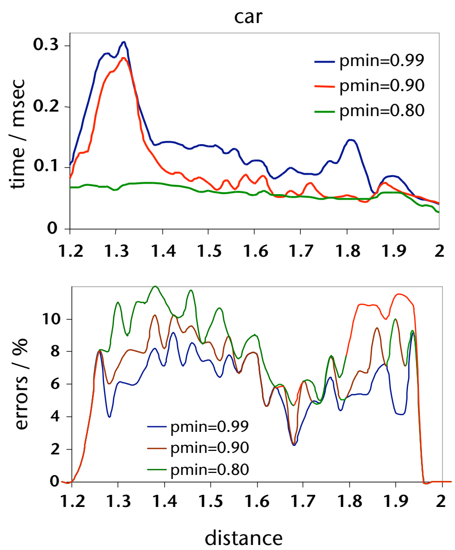
1. Partition A ∩ B with grid of s cells
2. Compute number of "well-filled" cells: s_A
3. Dito for B: s_B
4. Compute probability that x cells are "well-filled" from both A and B:



$$Pr[c(A \cap B) \geq x] = 1 - \sum_{t=0}^{x-1} \frac{\binom{s_B}{t}\binom{s-s_B}{s_A-t}}{\binom{s}{s_A}}$$

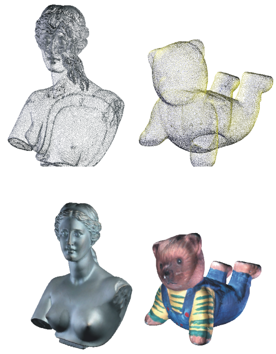


Results



Point Clouds

- Motivation: renaissance of points as object representation because of 3D scanners



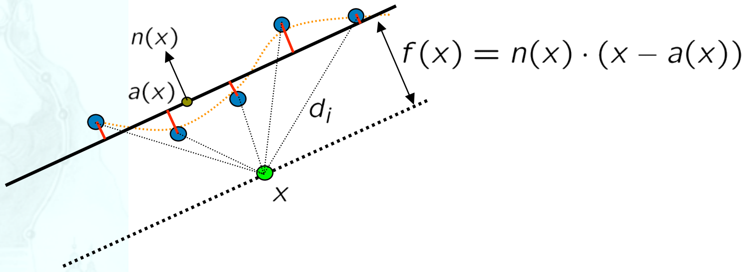
- Goal:
  - Fast collision detection between 2 given point clouds
  - No polygonal reconstruction

Definition of the surface

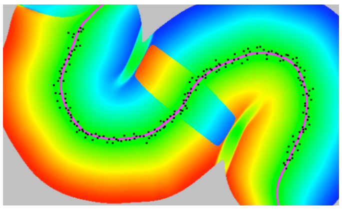
- Idea: assume "distance function" f from surface, then surface S is

$$S = \{x \in \mathbb{R}^3 \mid f(x) = 0\}$$

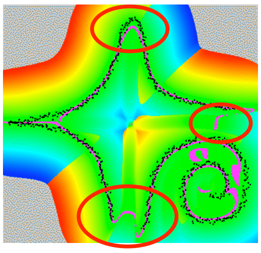
- "Distance" function f by Weighted Least Squares:



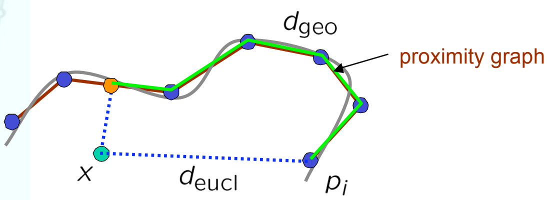
- Visualization $f(x)$ using Euclidean distance:



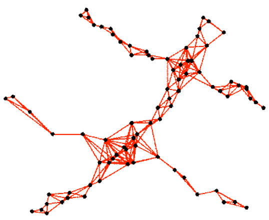
- Problems:



- Cause and solution:

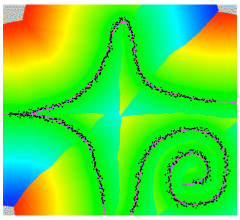


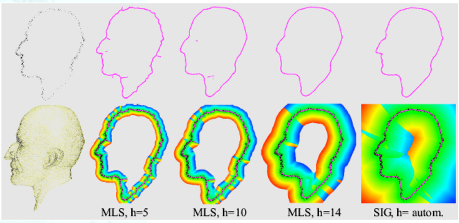
- Which neighborhood graph?
  → k-SIG (sphere-of-influence graph)



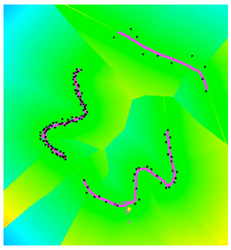
Benefits

- Much less artifacts
- Automatic, sampling-density independent detection of boundaries
- Automatic kernel bandwidth selection → handles different sampling densities automatically

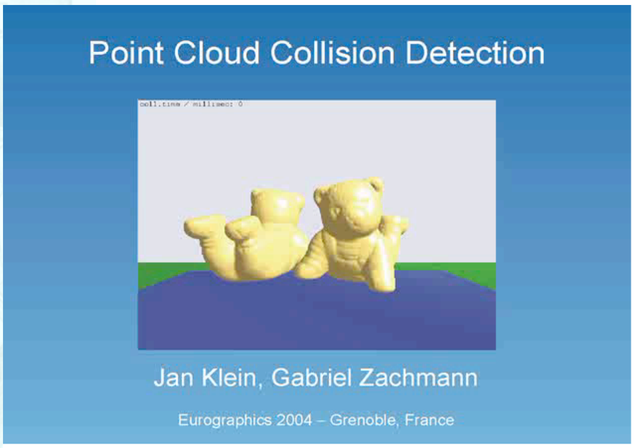




orig. PC — old method — our method

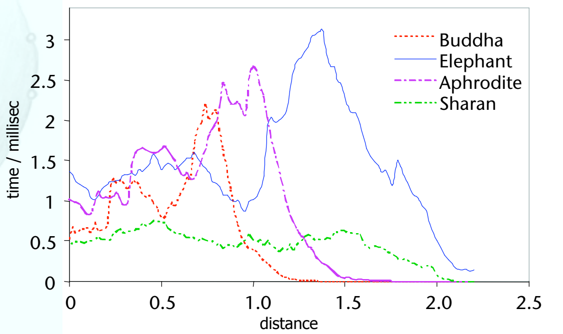


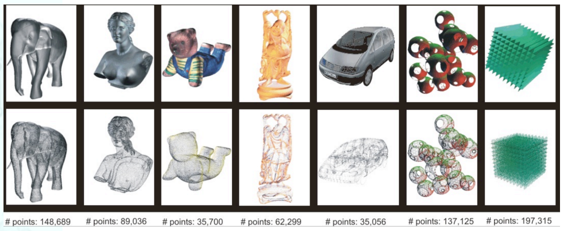
CD using Point Cloud Hierarchies



Motivation Abschn2 Abschn3 Conclusion

Results





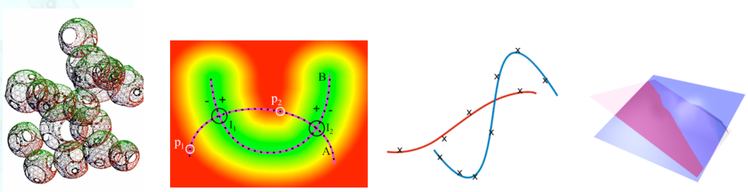
Coll.Det. of PCs using Stochastic Sampling

- Given two point clouds A and B (or subsets thereof), construct a sampling of

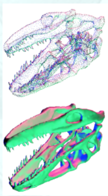
$$\mathcal{Z} = \{x \mid f_A(x) = f_B(x) = 0\}$$

- Overall method:

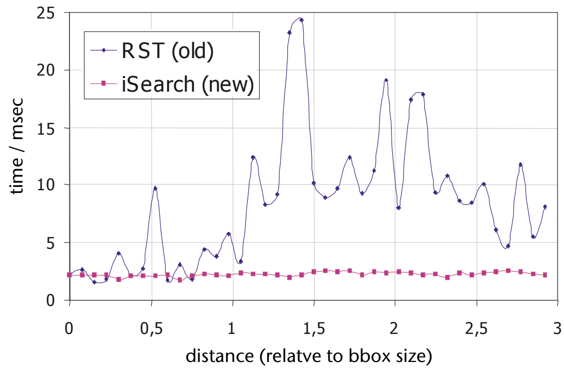
A,B → $(p_i, p_j) \in A$ on different sides of B → Approx. intersection points → Refined intersection point



Results



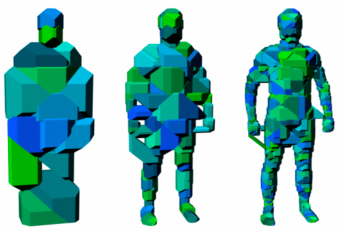
28,000 points



- Theoretical complexity: $O(\log \log N)$

Kinetic Bounding Volume Hierarchies

- For collision detection of deformable objects ...
- ... but not just for collision detection!
  - Can be applied to ray-tracing, occlusion culling, etc.



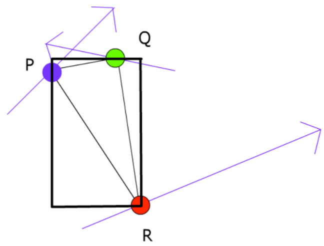
- Pre-processed hierarchy becomes invalid when object deforms

  → BVH must be rebuilt or updated after deformations

Our Approach

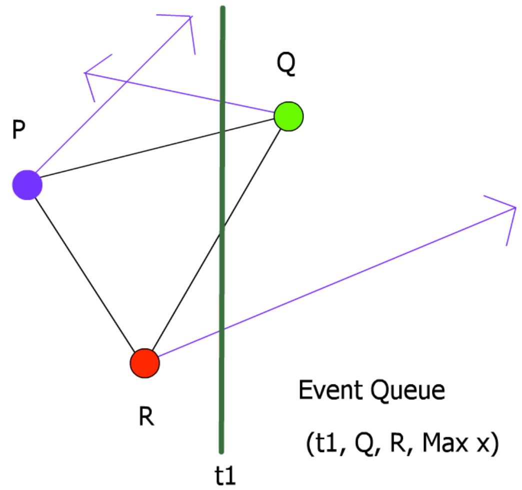
- Observation:
  - Motion in the physical world is normally continuous
  - Changes in the combinatorial structure of the BVHs occur only at discrete time points

→ We store only the combinatorial structure of the BVH and use an event-based approach for updating (kinetization)



Motivation Abschn2 Abschn3 Kinetization Conclusion

Kinetic Toy Example



Max
x Q
y Q

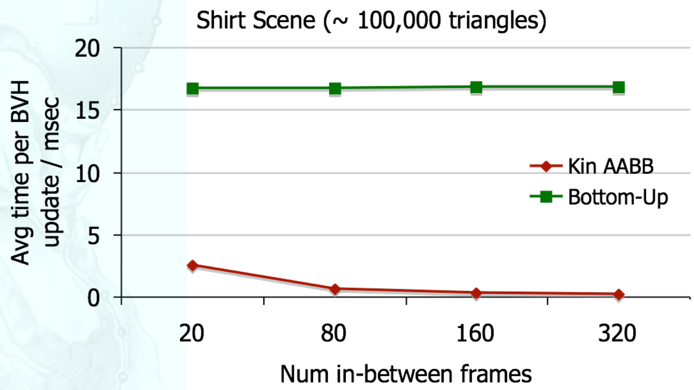
Min
x P
y R

Event Queue

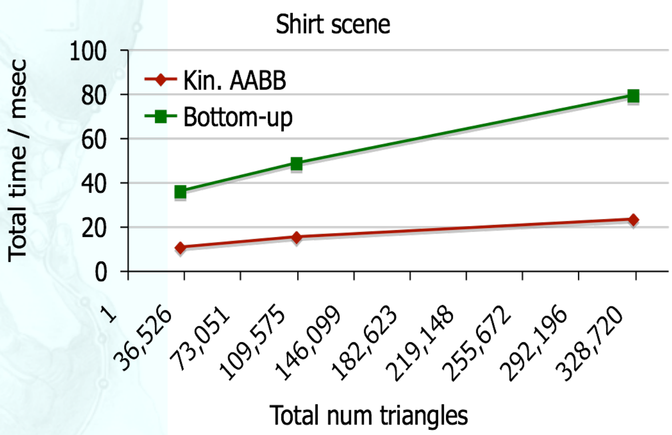
(t1, Q, R, Max x)

Motivation Abschn2 Abschn3 Kinetization Conclusion

Results

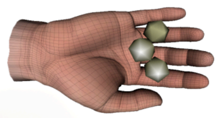


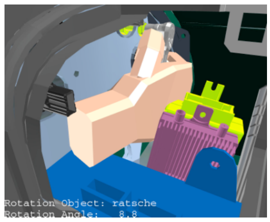
Total time incl. collision detection time



Natural Interaction

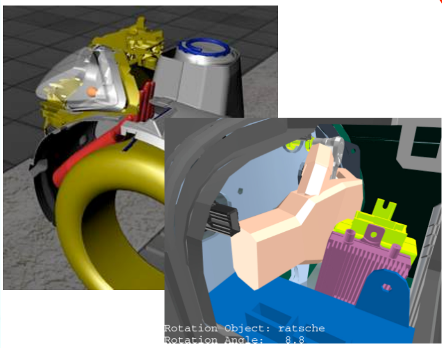
- Direct manipulation is more intuitive and sometimes even more efficient
- Goal:
  - Model and simulate the real human hand
  - Interaction between virtual environment and virtual hand
  - Not necessarily physically correct but physically plausible
- Applications:
  - Virtual assembly Simulation
  - 3D Sketching
  - Medical surgery training





Projekt: Natürliche Objekt-Manipulation

- Plausible, realitätsnahe, Echtzeit-Simulation
- DFG und EU-Antrag (Integrated Project)

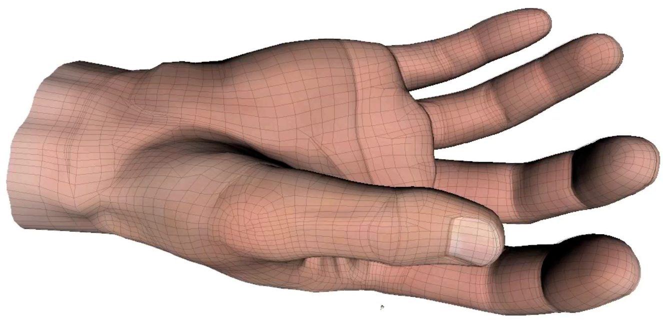


Implementation

- 17k quad mesh hand model
- Skeletal representation
- OpenSG for visualization
- Data Collection with VRJuggler
- Physical simulation by OpenDE
- Spring model for virtual grasping
- Does not rely on heuristics to estimate user intend or grasp state

Motivation Abschn2 Abschn3 Conclusion

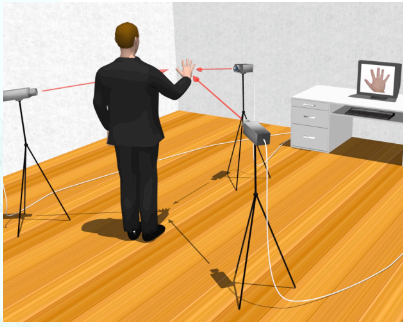
Result (work in progress)



Motivation Abschn2 Abschn3 Conclusion

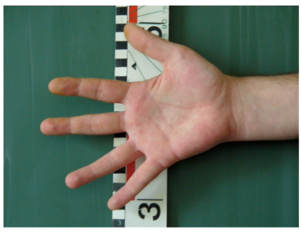
Real-Time Camera-Based 3D Hand Tracking

- Goals
  - Observe hand with cameras
  - Determine global hand position and orientation in 3d-space
  - Determine hand state, i.e. angles between fingers



Challenges

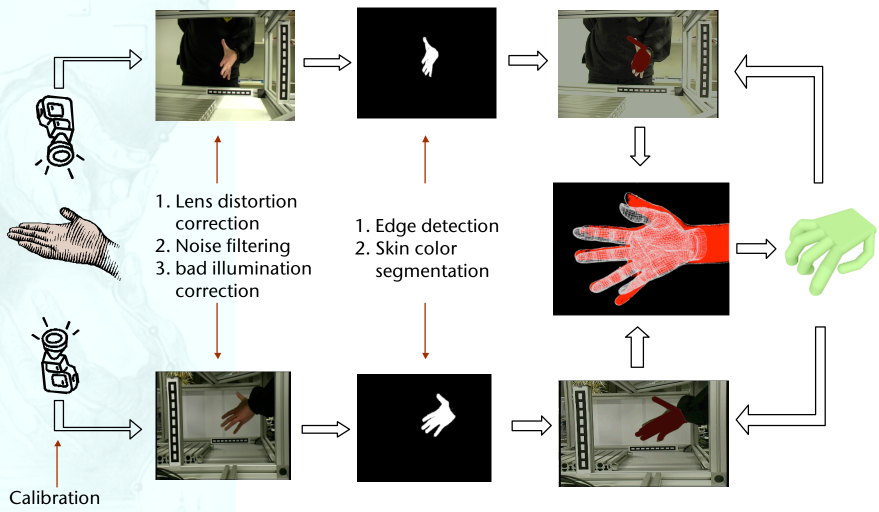
- Measurement noise
- Camera lens distortion
- Uncontrolled illumination
- Mutual occlusions of the hand
- Large working volume
- Fast hand motion
- High problem dimensionality (~ 27 DOFs)



Approach

- Capture hand with cameras from several viewpoints
- Preprocessing of captured images (correction of camera lens distortion, noise filtering)
- Edge detection
- Skin segmentation (bad illumination correction, skin region detection)
- Generate hand model (cylinders, triangle-mesh)
- Predict hand state
- Match hand model with image edges and skin regions in images

Overview of the System



Summary