Fast Pose Estimation with Parameter-Sensitive Hashing

Greg Shakhnarovich[⊛], Paul Viola[⋆], Trevor Darrell[⊛]

⊛ MIT Computer Science and Artificial Intelligence Lab

⋆ Microsoft Research

October 9, 2003

The problem

- The problem: recover articulated pose parameters (joint angles) from a single monocular image.

- Fitting a model is difficult: ill-posed problem, computationally extensive iterative process.

The problem

- The problem: recover articulated pose parameters (joint angles) from a single monocular image.
- Fitting a model is difficult: ill-posed problem, computationally extensive iterative process.
- But: the **local** relationship is often easy to model
  - Locally weighted regression (LWR)

$\Longrightarrow$ Need to find the neighbors (similar examples).

Related work

- Much work on tracking, less on estimation from a single frame.
- Usually uses point correspondece, feature detectors etc.
- Example-based methods:
  - [Athitsos and Sclaroff, CVPR 2003] shape classification; uses exact 1-NN search.
  - [Mori and Malik, ECCV 2002] "shape context"; uses contour, feature points.

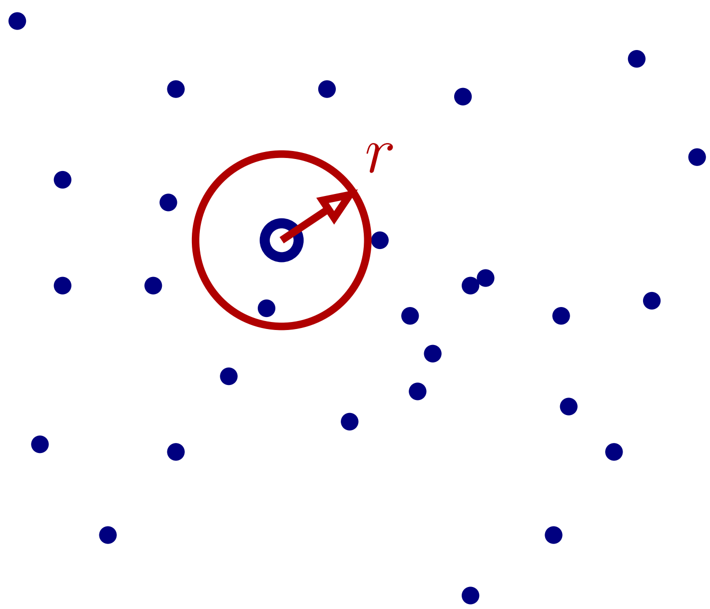
Is it computationally feasible?

- For complex parameter space (e.g. articulated pose), might need many ($10^5 - 10^6$) examples.
- Feature space is very high-dimensional.
- Exact similarity search algorithms (e.g. kd-trees, SR-trees...) suffer from "curse of dimensionality".

Is it computationally feasible?

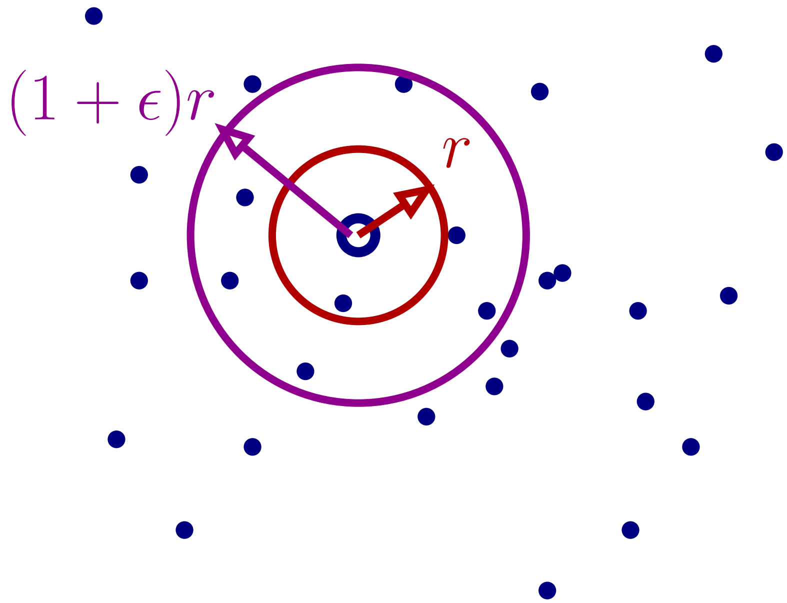
- For complex parameter space (e.g. articulated pose), might need many ($10^5 - 10^6$) examples.
- Feature space is very high-dimensional.
- Exact similarity search algorithms (e.g. kd-trees, SR-trees...) suffer from "curse of dimensionality".
- Randomized algorithms for similarity search to the rescue.

Locality-sensitive hashing



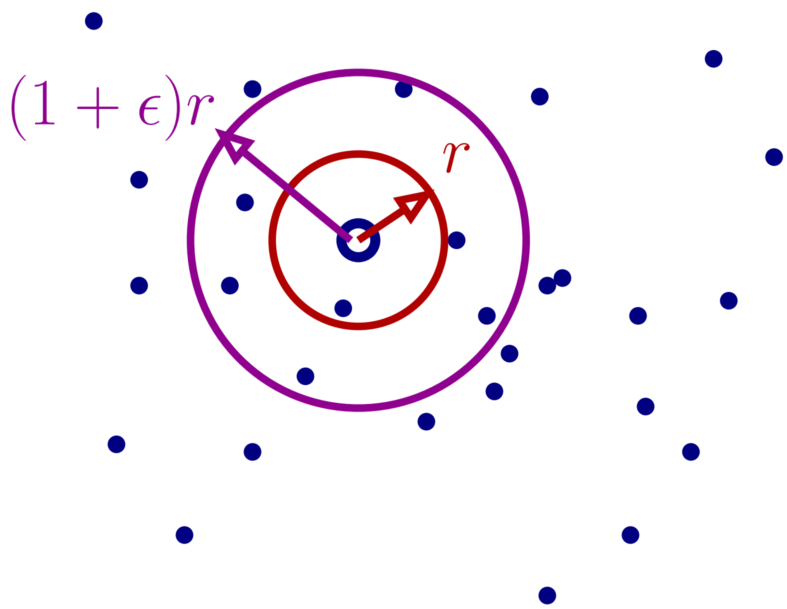
- r-neighbor

Locality-sensitive hashing



- r-neighbor
- $\epsilon - r$-neighbor

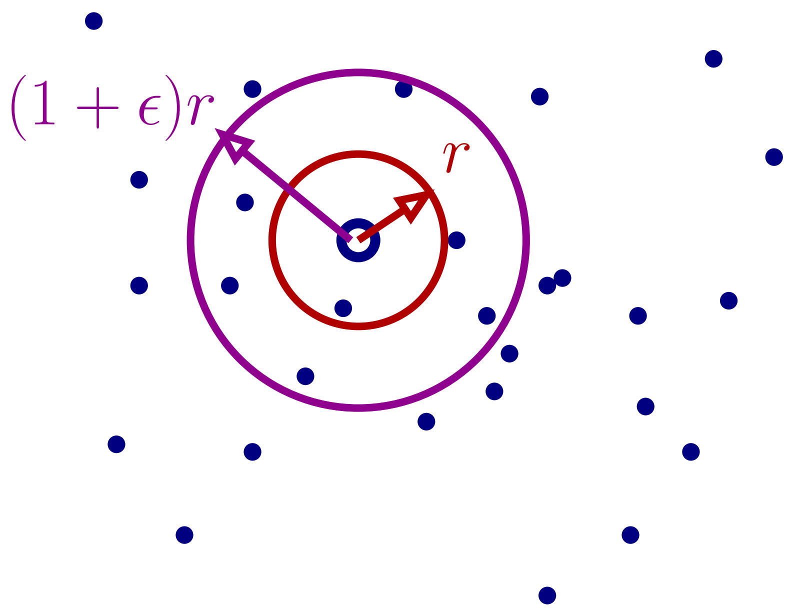
Locality-sensitive hashing



- r-neighbor
- $\epsilon - r$-neighbor

- LSH [Indyk & Motwani, '98-00]: solves the $\epsilon - r$-neighbor problem in $O\left(dn^{1/(1+\epsilon)}\right)$ (d – dimension, N – number of examples).

Locality-sensitive hashing



- r-neighbor
- $\epsilon - r$-neighbor

- LSH [Indyk & Motwani, '98-00]: solves the $\epsilon - r$-neighbor problem in $O\left(dn^{1/(1+\epsilon)}\right)$ (d – dimension, N – number of examples).
  - For $\epsilon = 1$ the running time is $O(d\sqrt{N})$
  - $N = 10^6$: speedup factor of 1000.

Locality-sensitive functions

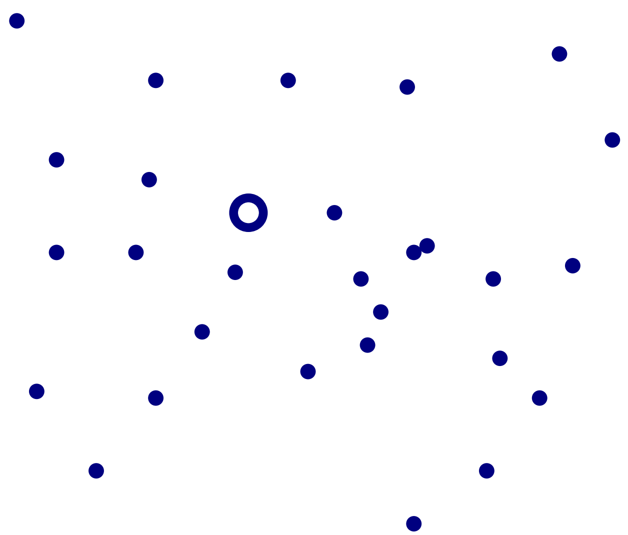
- Use a **locality-sensitive** family of binary functions h
- Probability of collision (same bit value) for two examples depends on the distance:
- If $d(\mathbf{x}, \mathbf{y}) < r$ (good collision) - high probability;
- If $d(\mathbf{x}, \mathbf{y}) > (1 + \epsilon)r$ (bad collision) low probability.

LSH

- Using l indpendent k-bit locality-sensitive hash functions

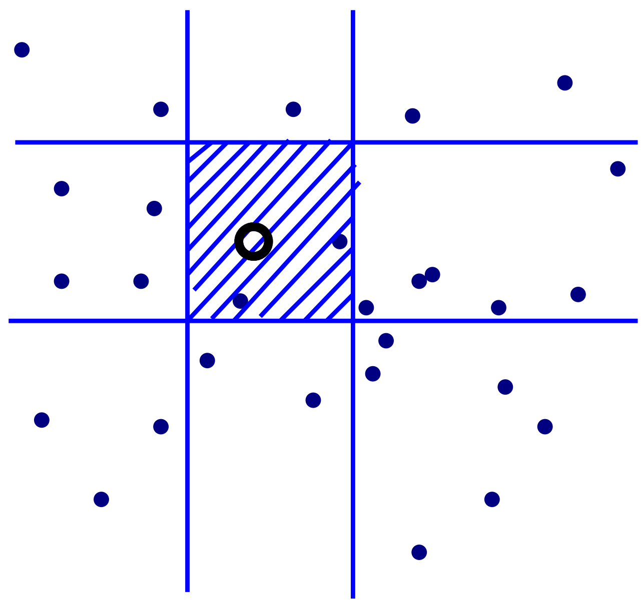
LSH

- Using l indpendent k-bit locality-sensitive hash functions:



LSH

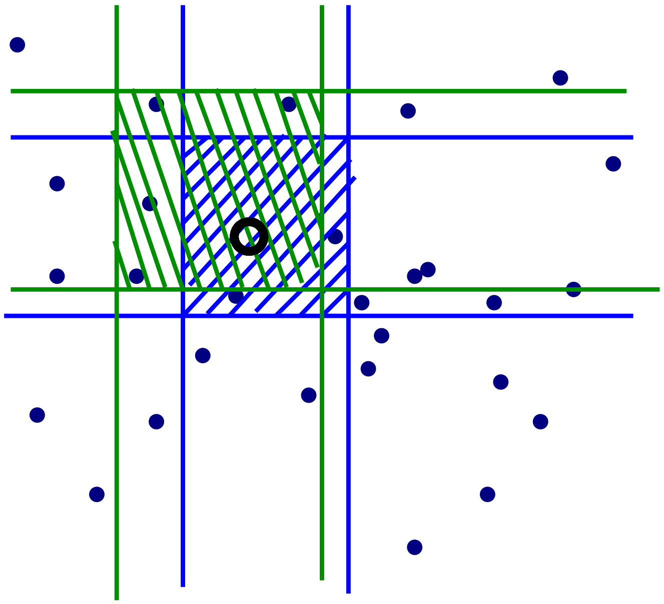
- Using l indpendent k-bit locality-sensitive hash functions:



- For each table find examples that fall into the same bucket with the input.

LSH

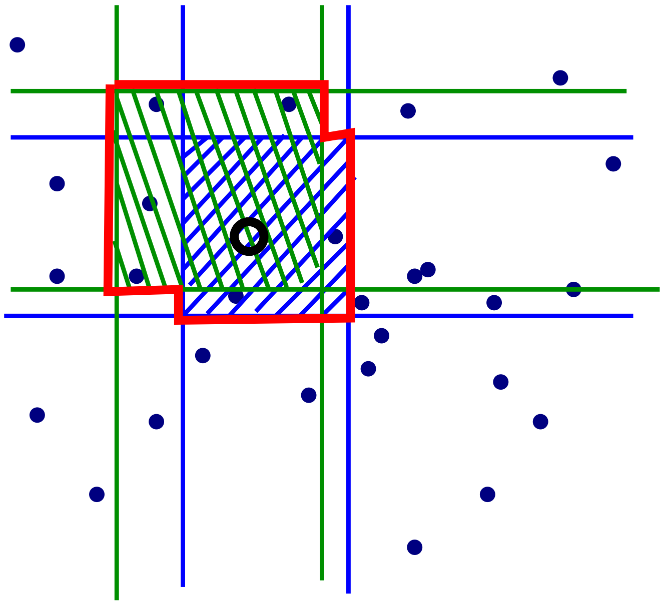
- Using l indpendent k-bit locality-sensitive hash functions:



- For each table find examples that fall into the same bucket with the input.

LSH

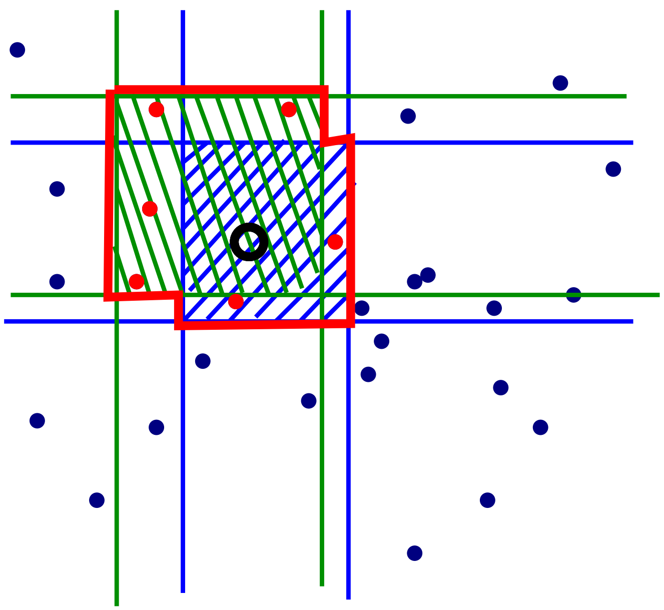
- Using l indpendent k-bit locality-sensitive hash functions:



- For each table find examples that fall into the same bucket with the input.
- Search the union of these buckets.

LSH

- Using l indpendent k-bit locality-sensitive hash functions:



- For each table find examples that fall into the same bucket with the input.
- Search the union of these buckets.

- With high probability, the union is small and contains good examples.

Similarity in parameter space

- LSH relies on distance in the feature space.
- We want similarity w.r.t. the distance d_θ **in the parameter values**.
- We don't know which hash functions are sensitive to the parameters.

Similarity in parameter space

- LSH relies on distance in the feature space.
- We want similarity w.r.t. the distance d_θ **in the parameter values**.
- We don't know which hash functions are sensitive to the parameters.
- Instead, we will estimate the performance of the binary hash functions empirically, and select a locality-sensitive subset with respect to d_θ.
  $\Longrightarrow$ **P**arameter-**s**ensitive **h**ashing.

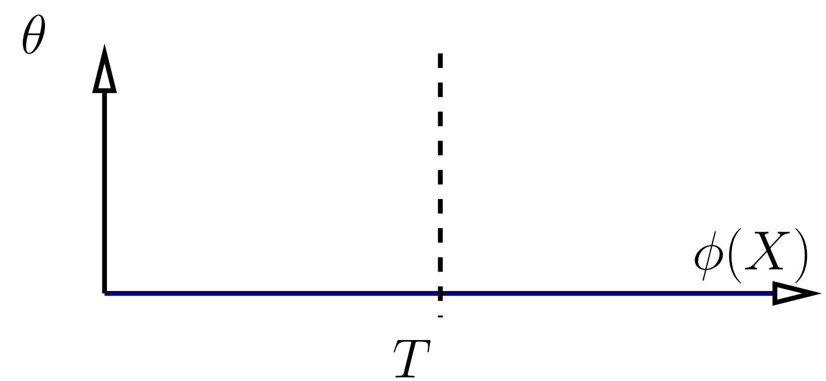
Hashing and classification

- A **paired example** $\langle(\mathbf{x}_i, \theta_i), (\mathbf{x}_j, \theta_j)\rangle$ is labeled:
  - Positive if $d_\theta(\theta_i, \theta_j) < r$;
  - Negative if $d_\theta(\theta_i, \theta_j) > (1+\epsilon)r$.

Hashing and classification

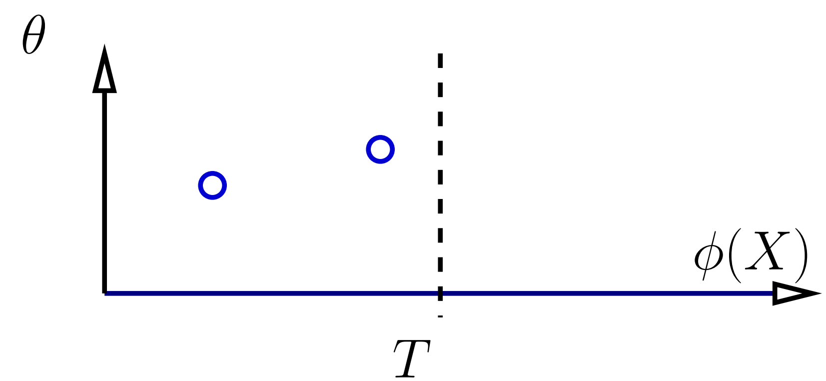
- A **paired example** $\langle(\mathbf{x}_i, \theta_i), (\mathbf{x}_j, \theta_j)\rangle$ is labeled:
  - Positive if $d_\theta(\theta_i, \theta_j) < r$;
  - Negative if $d_\theta(\theta_i, \theta_j) > (1+\epsilon)r$.

- $h_{\phi,T}$ applied on a paired examples will either:



Hashing and classification

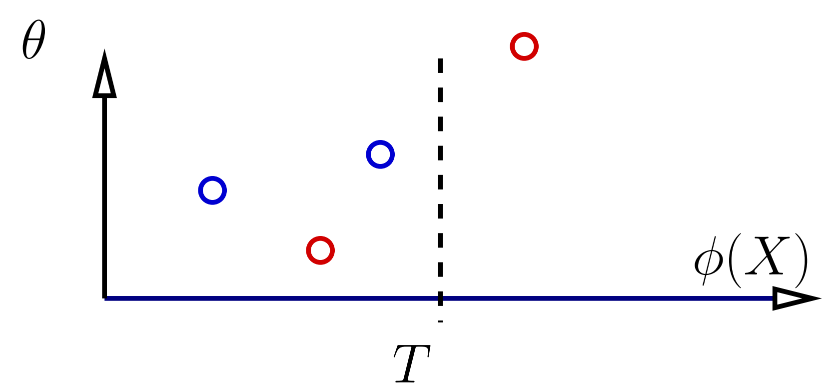
- A **paired example** $\langle(\mathbf{x}_i, \theta_i), (\mathbf{x}_j, \theta_j)\rangle$ is labeled:
  - Positive if $d_\theta(\theta_i, \theta_j) < r$;
  - Negative if $d_\theta(\theta_i, \theta_j) > (1+\epsilon)r$.
- $h_{\phi,T}$ applied on a paired examples will either:
  - Place both in the same bin



Hashing and classification

- A **paired example** $\langle(\mathbf{x}_i, \theta_i), (\mathbf{x}_j, \theta_j)\rangle$ is labeled:
  - Positive if $d_\theta(\theta_i, \theta_j) < r$;
  - Negative if $d_\theta(\theta_i, \theta_j) > (1+\epsilon)r$.

- $h_{\phi,T}$ applied on a paired examples will either:
  - Place both in the same bin ,or
  - Separate between them.



Hashing and classification

- $h_{\phi,T}$ classifies the pair as positive if both components fall in the same bucket.
  - $\Pr(\text{Bad collision}) =$ false positive rate
  - $\Pr(\text{Good collision}) =$ true positive rate

- Selection mechanism:
  - Sample a large paired training set;
  - Set target values for false positive and false negative rates;
  - Select binary functions that meet the target.

Paired examples

AND

POS

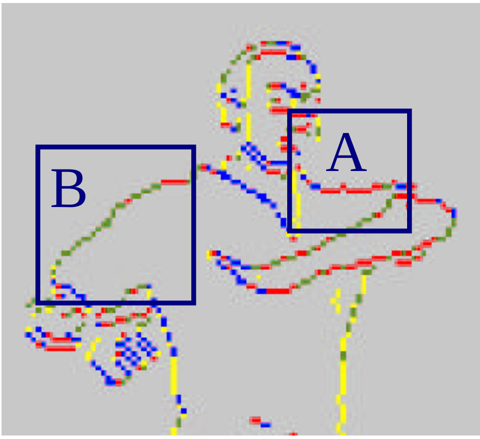
POS

NEG

NEG

Representation

- Image features: concatenated multi-scale edge direction histograms



$0 \quad \pi/4 \quad \pi/2 \quad 3\pi/4$

$\phi_{107}(\mathbf{x}) = \sum_A x_0$

$\phi_{7033}(\mathbf{x}) = \sum_B x_{\pi/4}$

- Binary functions (axis-parallel decision stumps)

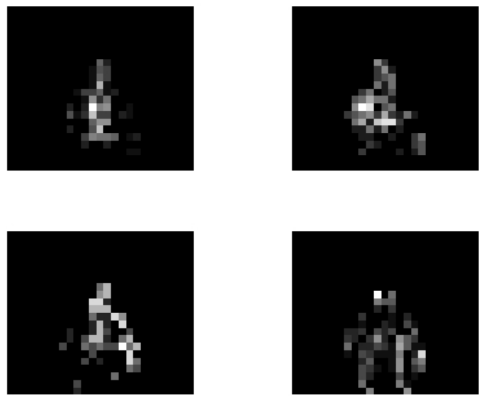
$$h_{\phi,T}(\mathbf{x}) = \begin{cases} +1 & \text{if } \phi(\mathbf{x}) \geq T, \\ -1 & \text{otherwise} \end{cases}$$

for a feature ϕ and a threshold T.

Pose with PSH

Input image

Compute features



Find similar examples

Robust LWR

Data collection

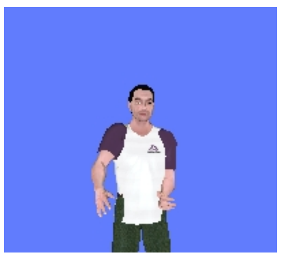
- Labeled data generated with POSER®.
- 150,000 images, 200$\times$180 pixels.
- 13 DOF (shoulders, elbows, collar bone, $+$ torso rotation).
- Nuisance parameters: illumination, head and hand pose, facial expression, clothing.

Paired training

- Hash function selection: 1,775,000 paired examples.
- Selected 137 out of 5,123 non-constant features.
  - 18-bit hash functions, 150 hash tables.
  - Without selection, would need 40 bits, 1000 hash tables.

Synthetic test set

- Test on 1,000 synthetic examples, to evaluate different methods.
- "Correct" estimate = mean error in angles less than 15 degrees.




- 1-NN: **33%** correct;
- 25-NN: **67%**;
- 25-NN, robust linear LWR **70%**.

Results on real data

- Office environment; simple background subtraction.
- No ground truth - evaluation by "eye-balling".
- MATLAB implementation: less than 1 sec per query.
- Only 2,000 candidates per query (80 times speedup).

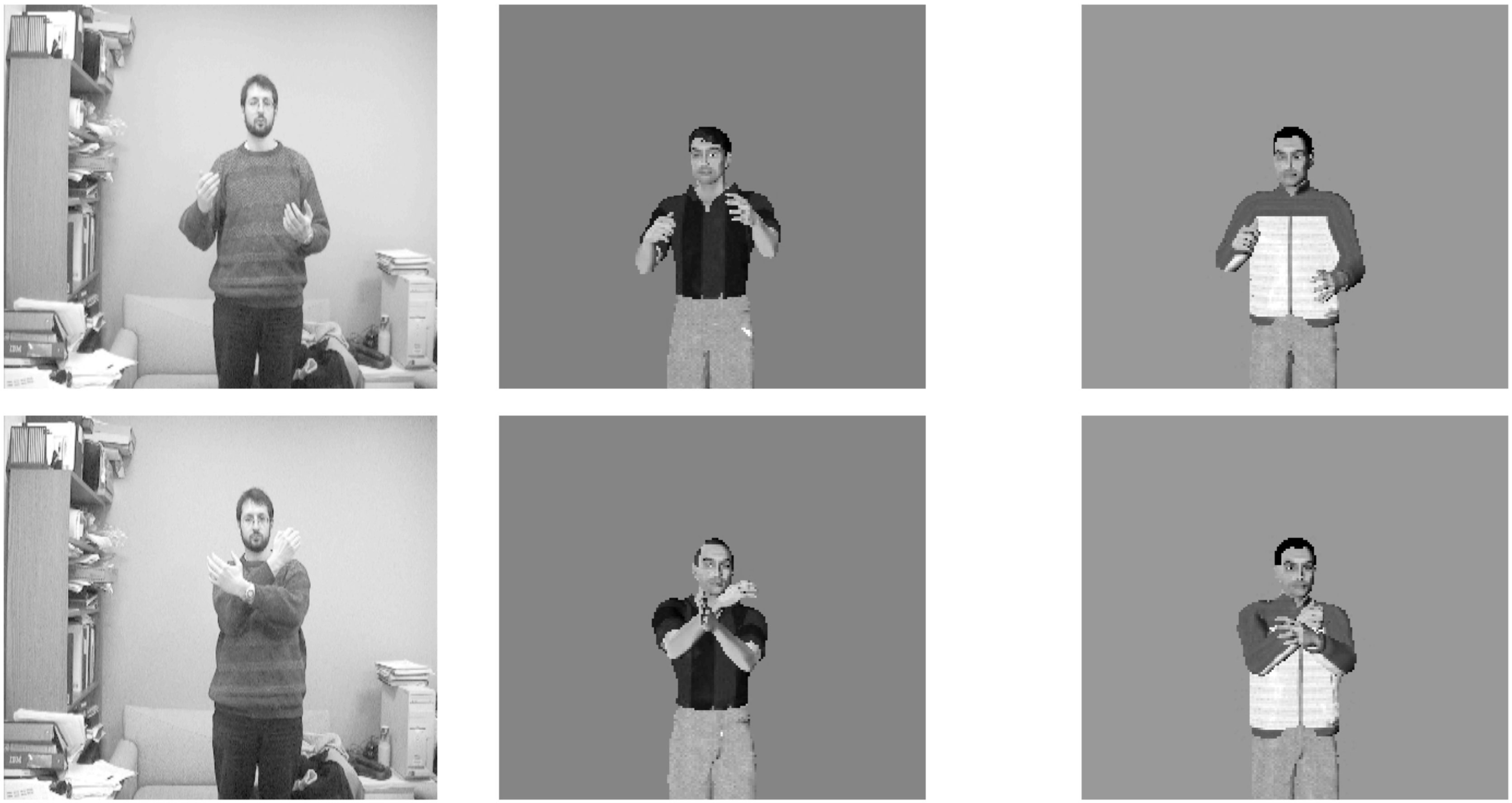
Results on real data

- Office environment; simple background subtraction.
- No ground truth - evaluation by "eye-balling".
- MATLAB implementation: less than 1 sec per query.
- Only 2,000 candidates per query (80 times speedup).

(No real data used in training!)

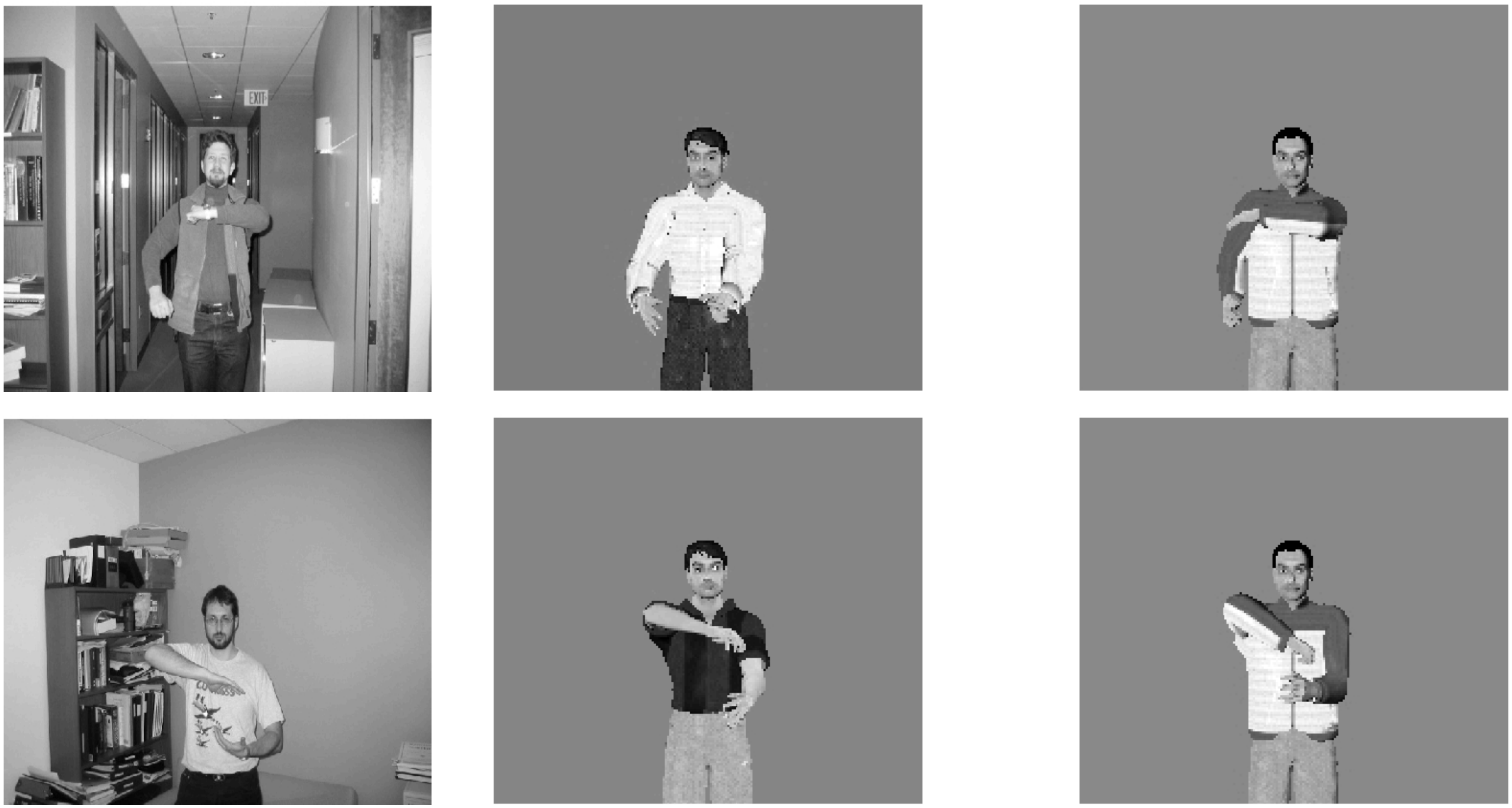
Results on real data

INPUT | 1-NN | ROBUST LWR - 12NN



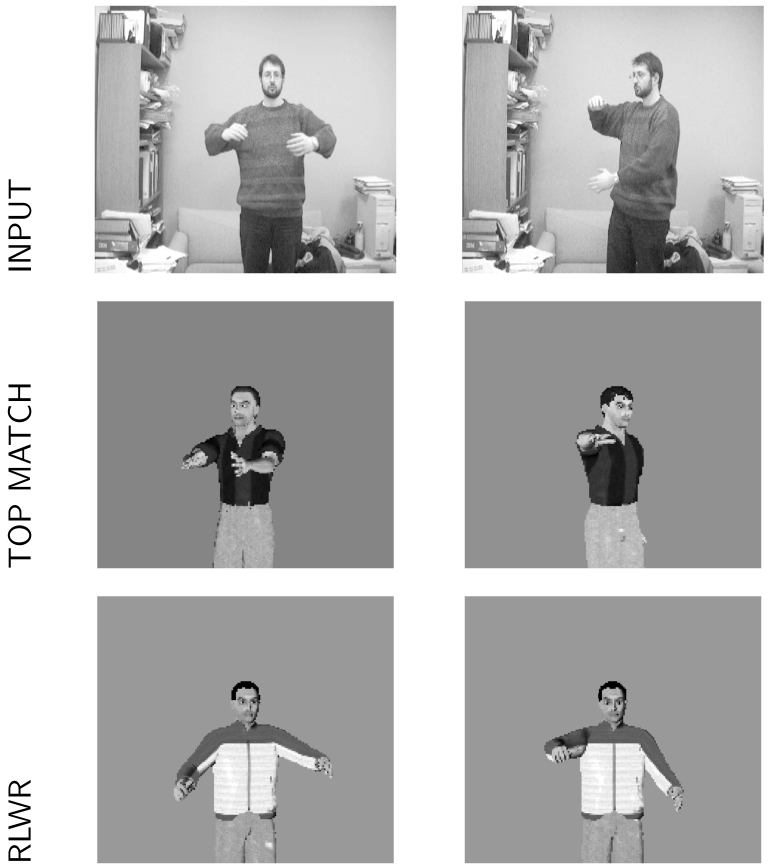
Results on real data

INPUT | 1-NN | ROBUST LWR - 12NN



More results

Failures



Future work

- Apply PSH to other estimation problem in vision where large amount of labeled data available.
  - Age estimation
  - Shape inference
  - ...
- Integrating temporal information (tracking?)
  - Treat estimated neighbors as "particles"?

Summary

- Randomized algorithms for similarity search make example-based methods feasible.
- Parameter-sensitive hashing for estimation.
- Paired classification paradigm for selecting hash functions.
- Articulated pose estimation from single frame using large synthetic corpus.

Questions?..