Compact Real-Time Modeling of Seated Humans by Video Sprite Sequence Quantization

Chun Jia

Voicu Popescu

Motivation

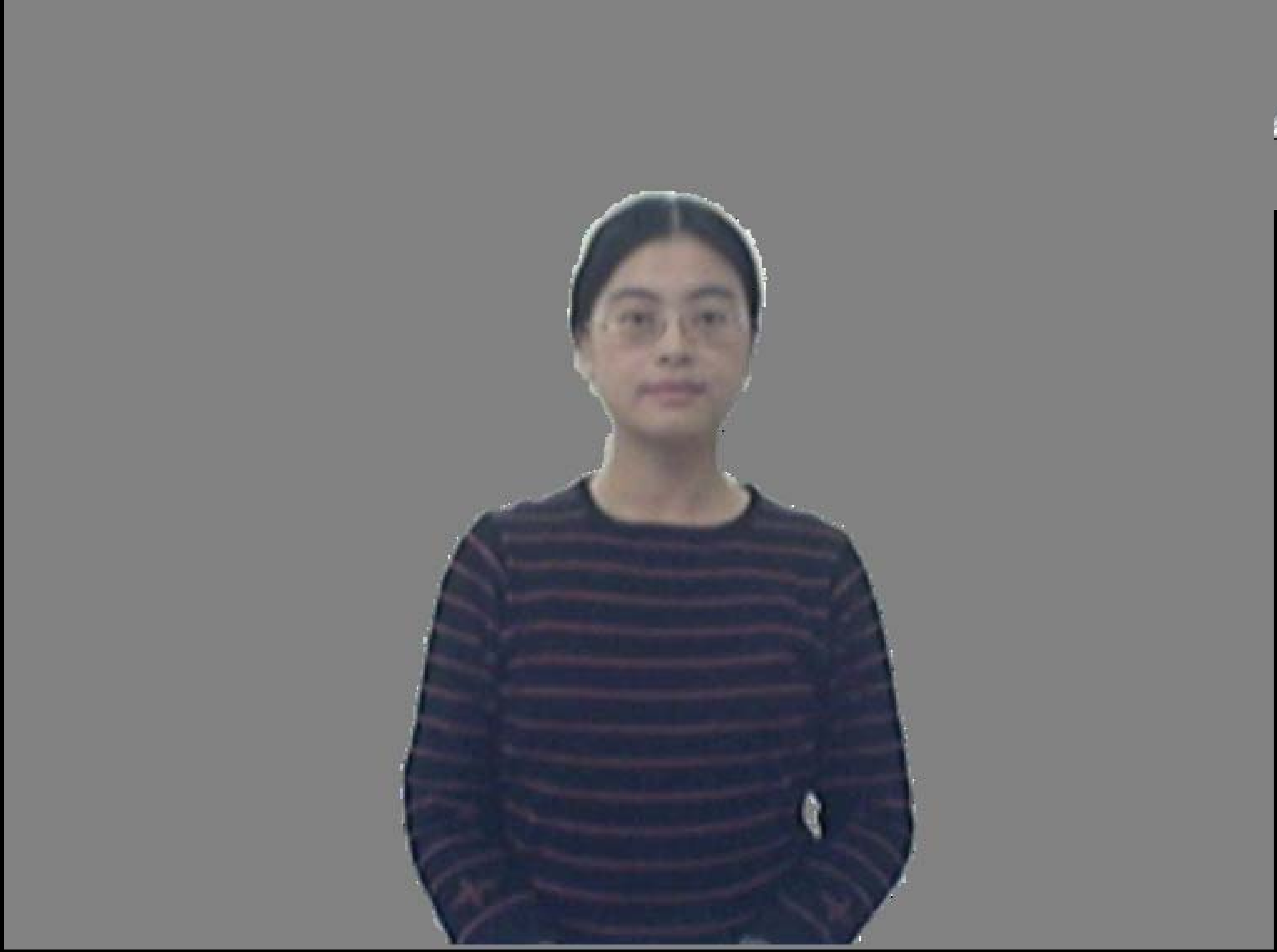
- Inadequate support for distributed multi-user computer graphics applications
  - Distance learning
  - Teleconferencing
  - Distributed virtual environments

Problems

- Modeling
  - Real-time modeling of participants is challenging (depth acquisition)
- Bandwidth
  - Low bandwidth between acquisition and rendering sites (1-3 Mbps for commodity connectivity such as DSL or cable modem)

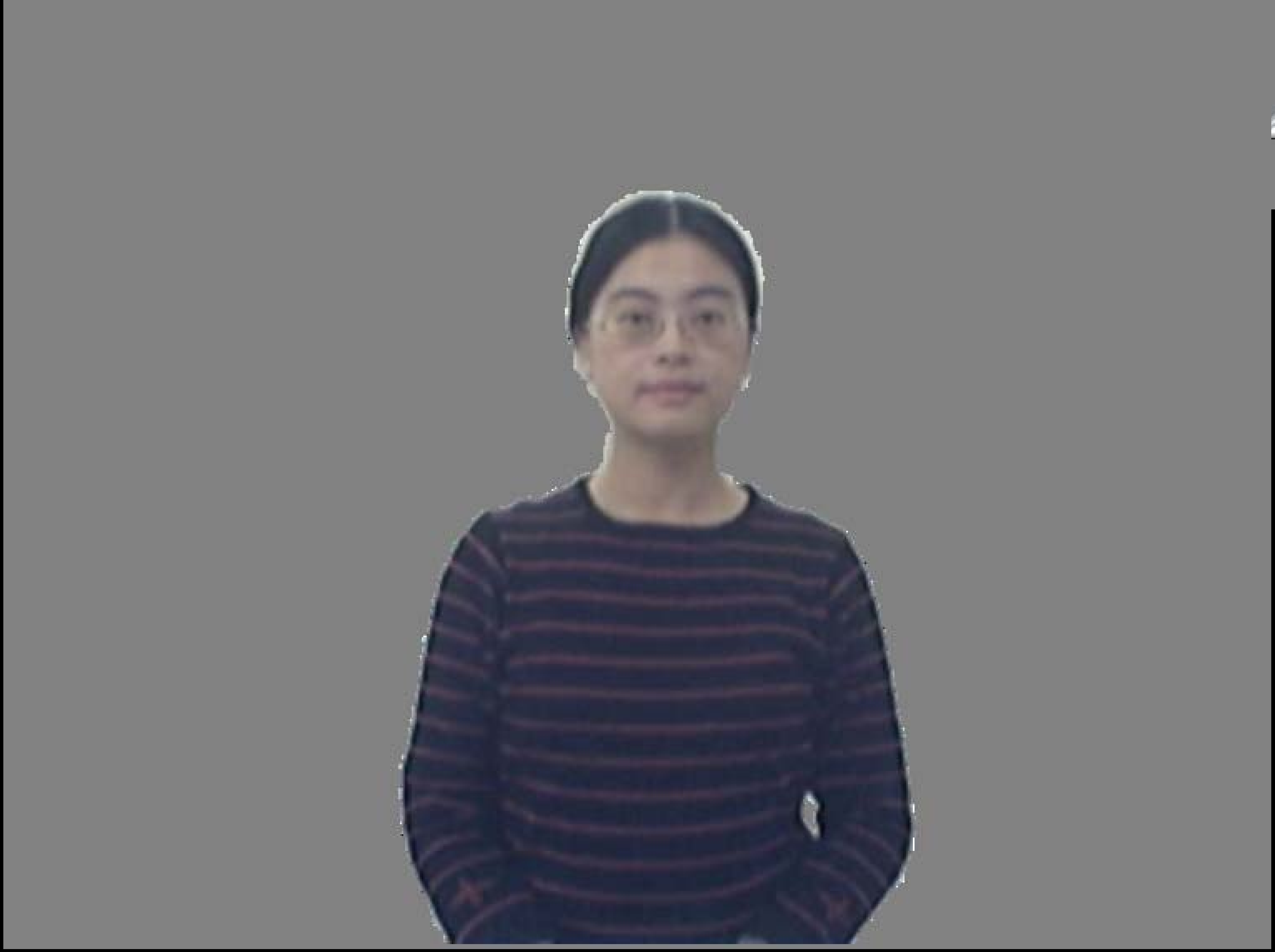
Approach

- Modeling: video sprites [SHADE 1996]
  - Inexpensive—$100 webcam
  - Robust and efficient—use favorable background
  - Effective—realism of video frames
  - Approximate—limited range of desired views
  - Data-intensive— 640x480 @10fps video sprite require 1-5 Mbps



MPEG compression high quality – 280 Kbits / sprite

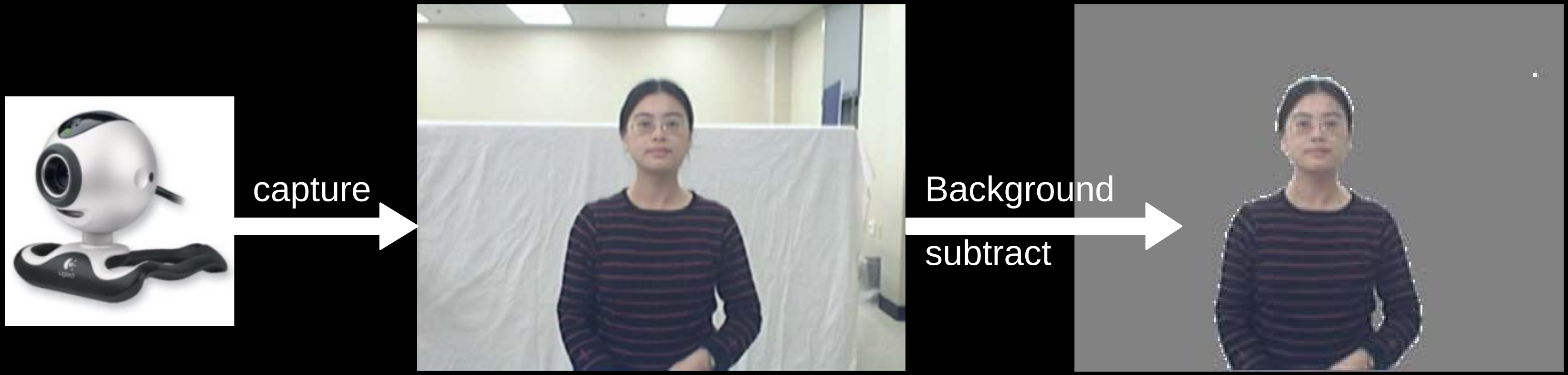
MPEG compression low quality – 53 Kbits / sprite



Our result – 16 bits / sprite

Approach

- Modeling: video sprites [SHADE 1996]



capture

Background subtract

Webcam

Video frame

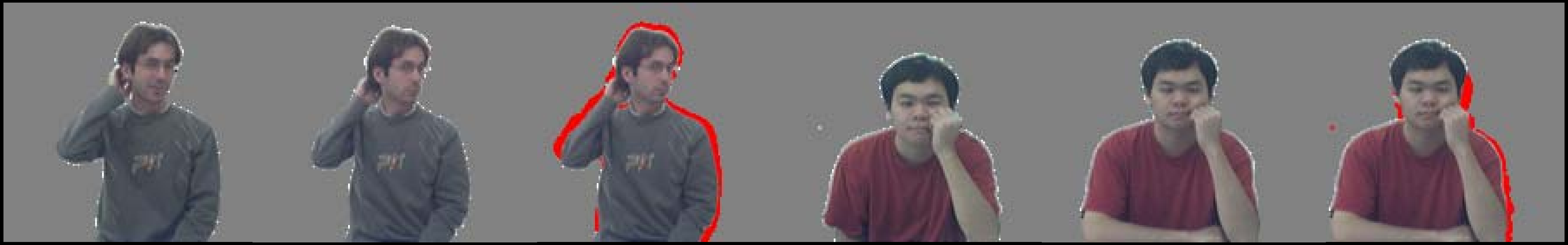
Video sprite

Approach

- Modeling: video sprites [SHADE 1996]
- Bandwidth reduction: leverage limited number of representative body poses for seated participant
  - □ Neutral, raising right hand, chin on hand …

Approach

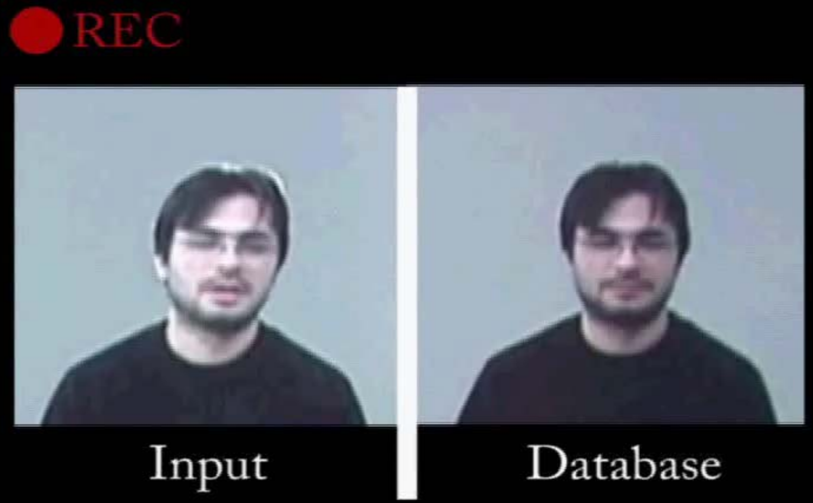
- Modeling: video sprites [SHADE 1996]
- Bandwidth reduction: leverage limited number of representative body poses for seated participant
  - □ Pixel-level differences do not necessarily imply semantic differences



Input sprite Database sprite difference Input sprite Database sprite difference

Approach

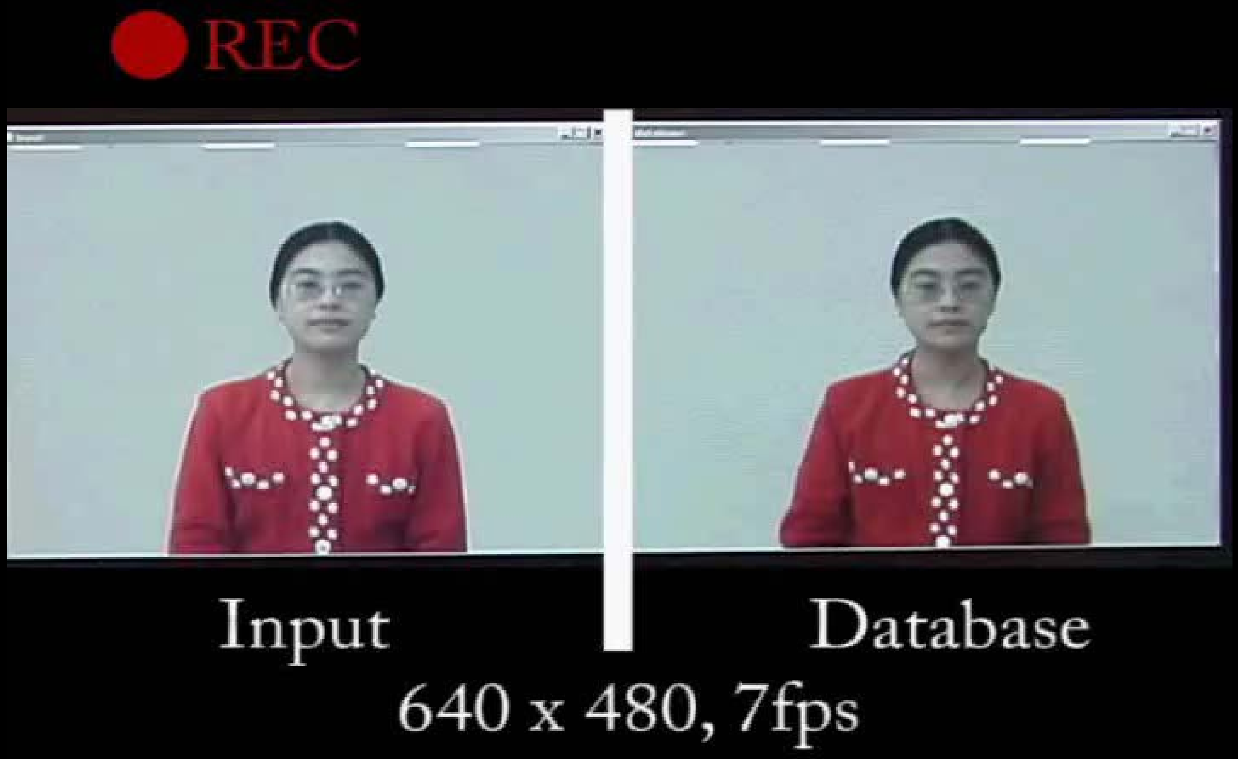
- Modeling: video sprites [SHADE 1996]
- Bandwidth reduction: leverage limited number of representative body poses for seated participant
  - □ Not all frame regions have same importance



Approach

- Modeling: video sprites [SHADE 1996]
- Bandwidth reduction: leverage limited number of representative body poses for seated participant
  - □ Video compression doesn’t!

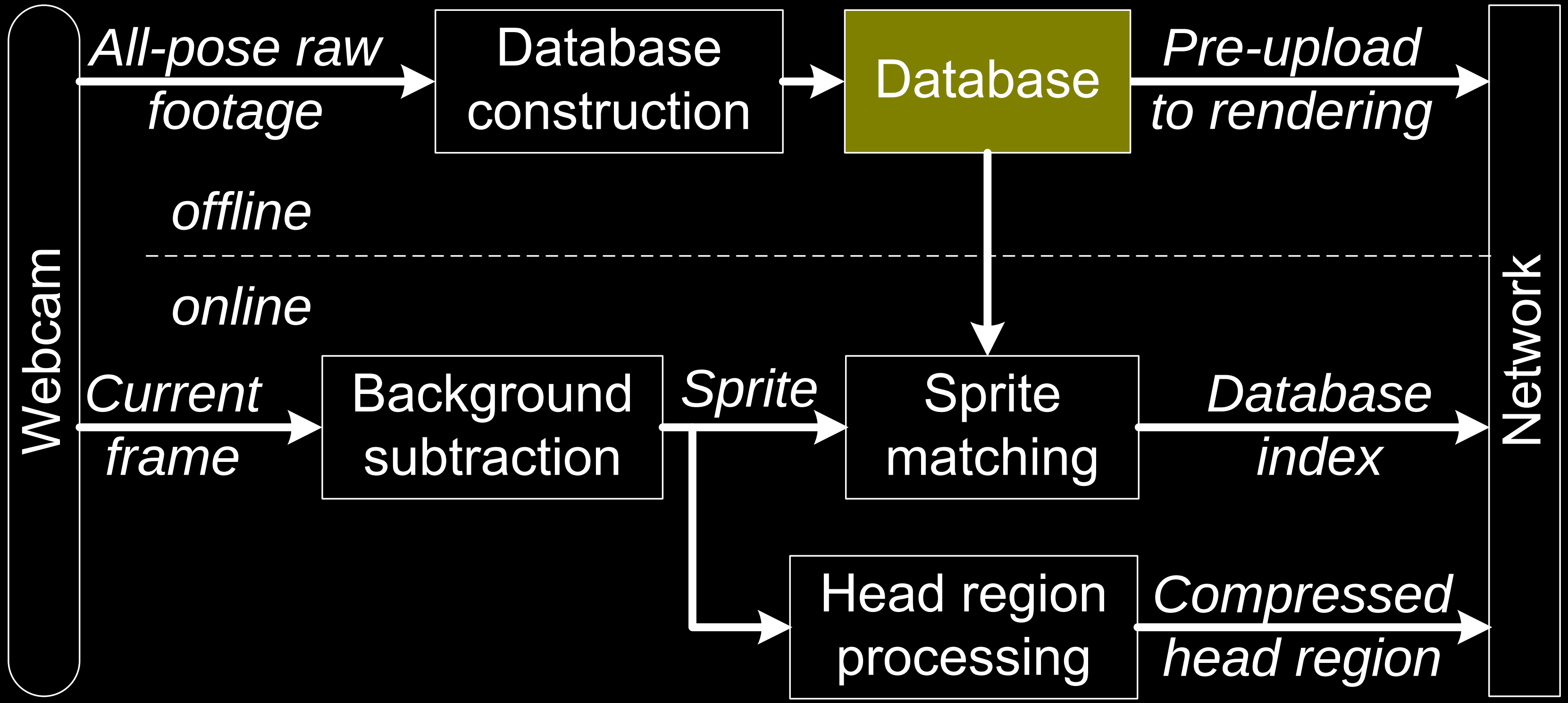
Results Preview



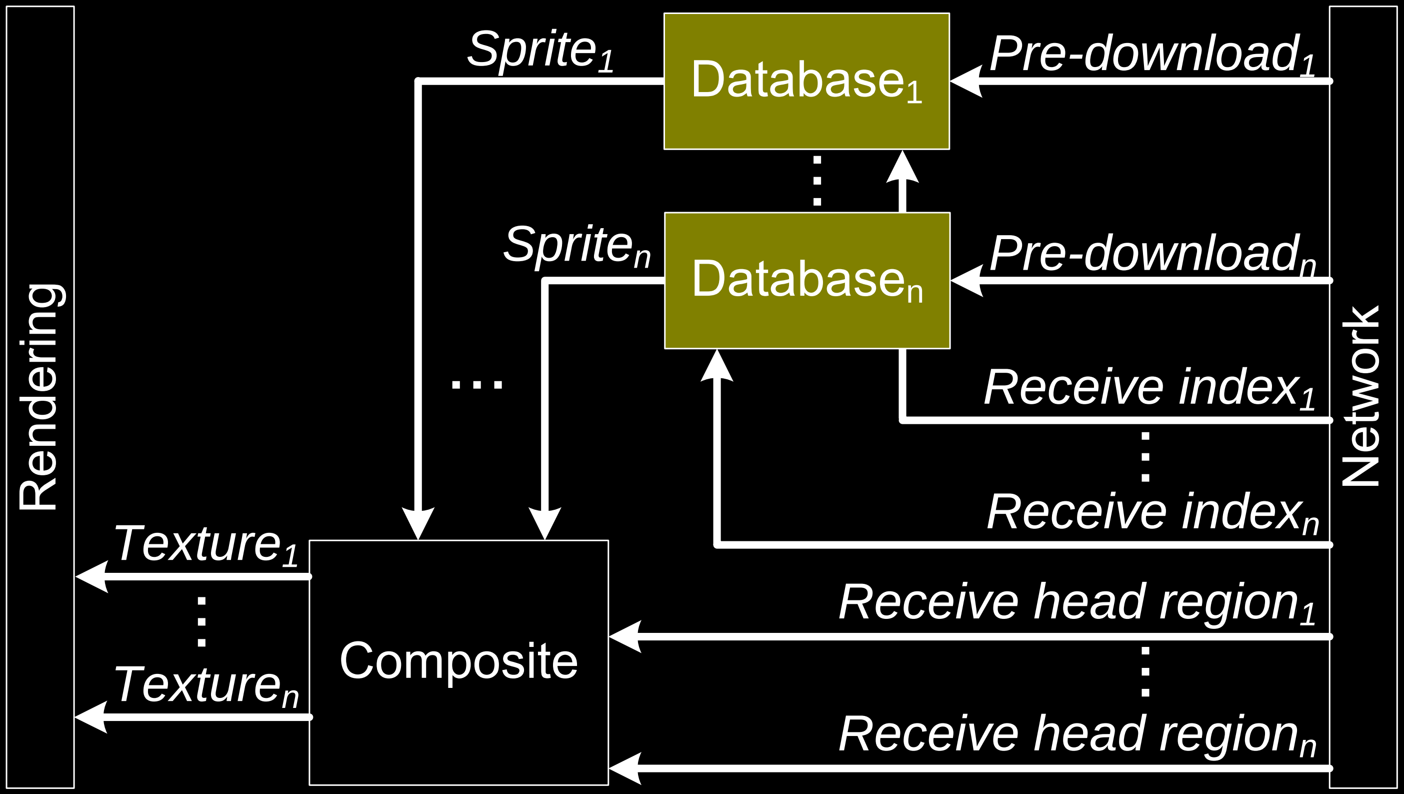
System Overview

- Acquisition Module
- Rendering Module

Acquisition Module



Rendering Module



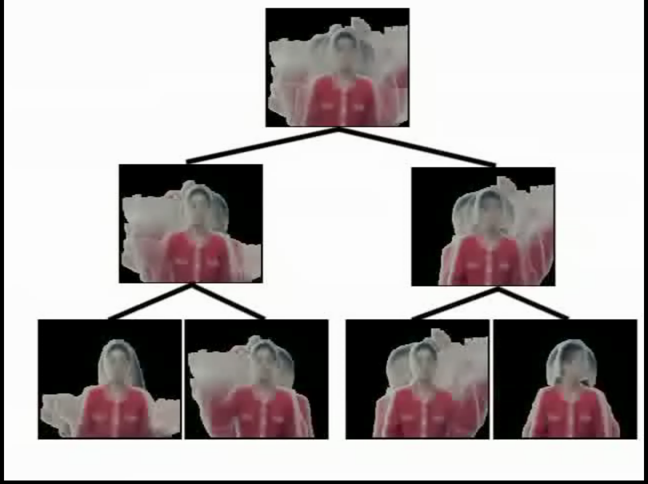
Database Construction

- Transform frame into sprite using background subtraction
- Segment the raw video footage into sequence by pauses between poses

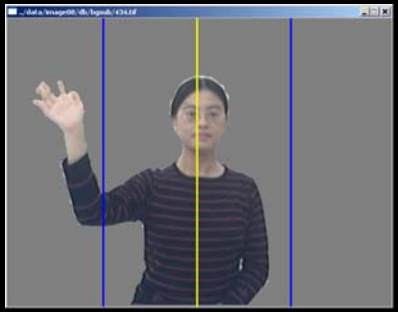
Off-Line Database Construction

Database Construction

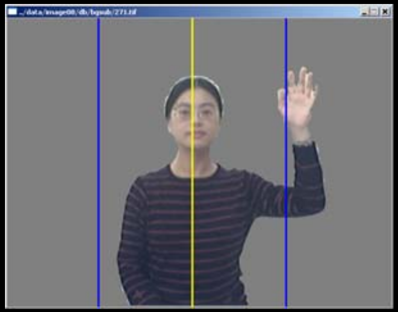
- Construct a binary tree of depth 3 on sprite sequence shape
  - □ Shape is classified for each sprite sequence
  - □ Tree is constructed recursively from the set of sprite sequence based on shape



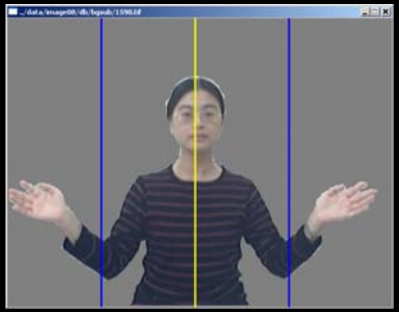
Shape classification



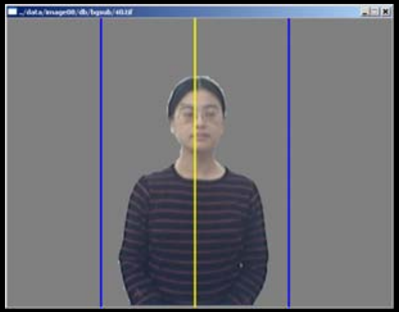
yes-left



yes-right



yes-left & yes-right

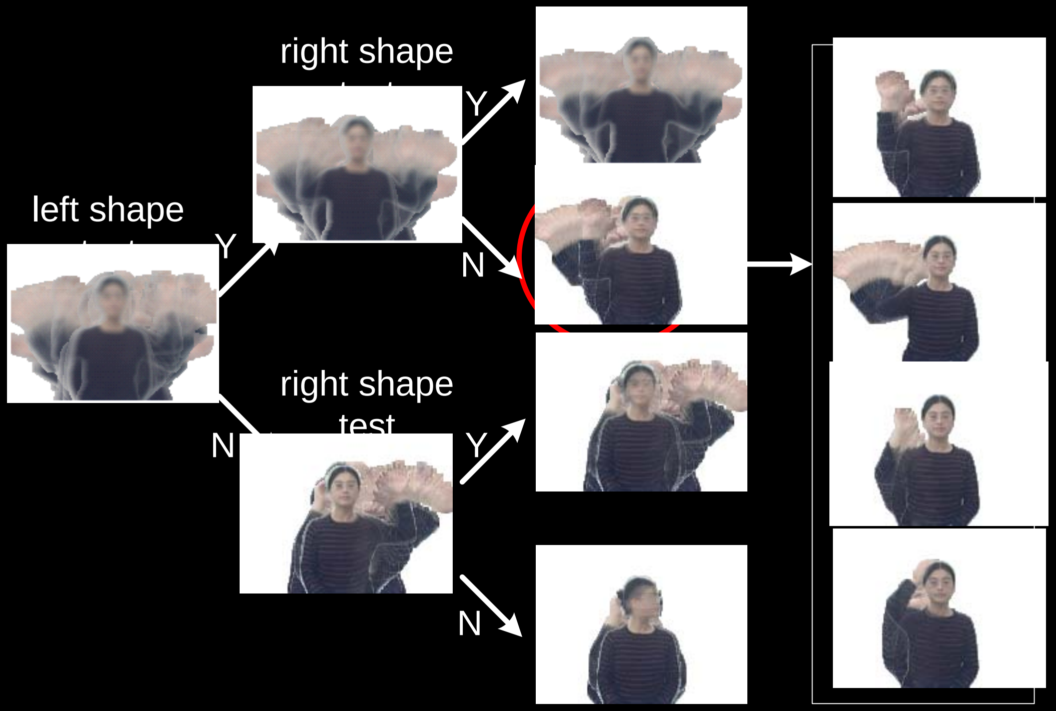


no-left & no-right

- Shape is a bitmap with pixel value 1 if it is hit by any sprite in the sequence, 0 otherwise
- Shoulder lines (blue) and central line (yellow) define the three regions
- Shape of sprite sequence is defined as left, right, left + right, others.

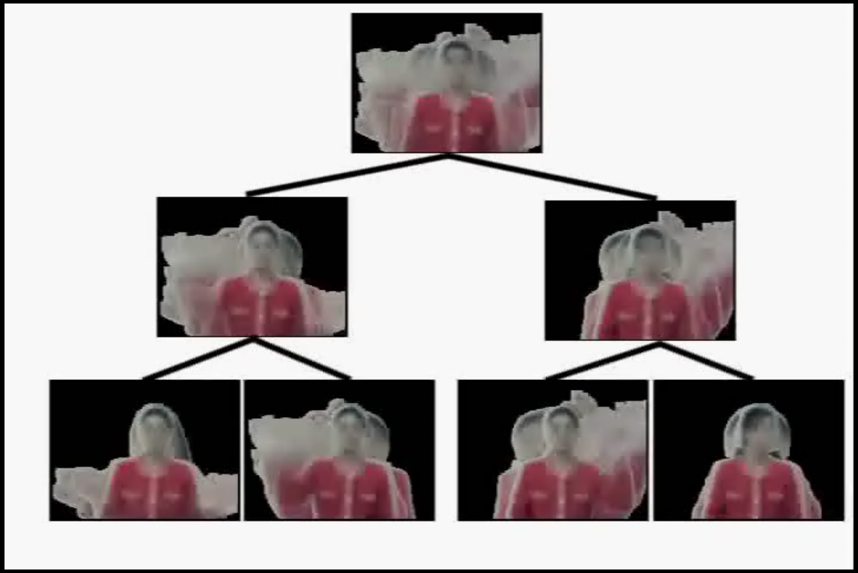
Tree construction

- Assignment of sprite sequence according to which region its shape crosses into
- A tree leaf stores an array of sprite sequence



- 2550 raw frames
- 32 sequences
- 1592 sprites
- 4 leaves (top -> bottom)
  - □ 10 sequences
  - □ 4 sequences
  - □ 10 sequences
  - □ 8 sequences

Tree construction

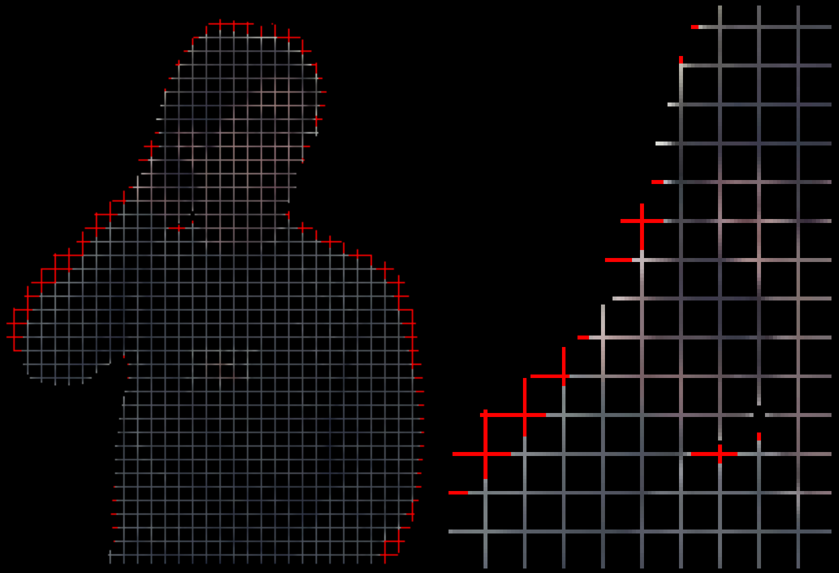


- 1650 raw frames
- 28 sequences
- 1044 sprites
- 4 leaves (left -> right)
  - □ 7 sequences
  - □ 7 sequences
  - □ 7 sequences
  - □ 7 sequences

Real-time Sprite Matching

- For each input sprite, down-sample and blur the sprite
- Traverse the tree based on sprite shape until appropriate leaf of the tree is found
- Discard leaf sprite sequences with unmatched shape
- Linear searching the remaining leaf sprite sequences to get the best match based on color + shape

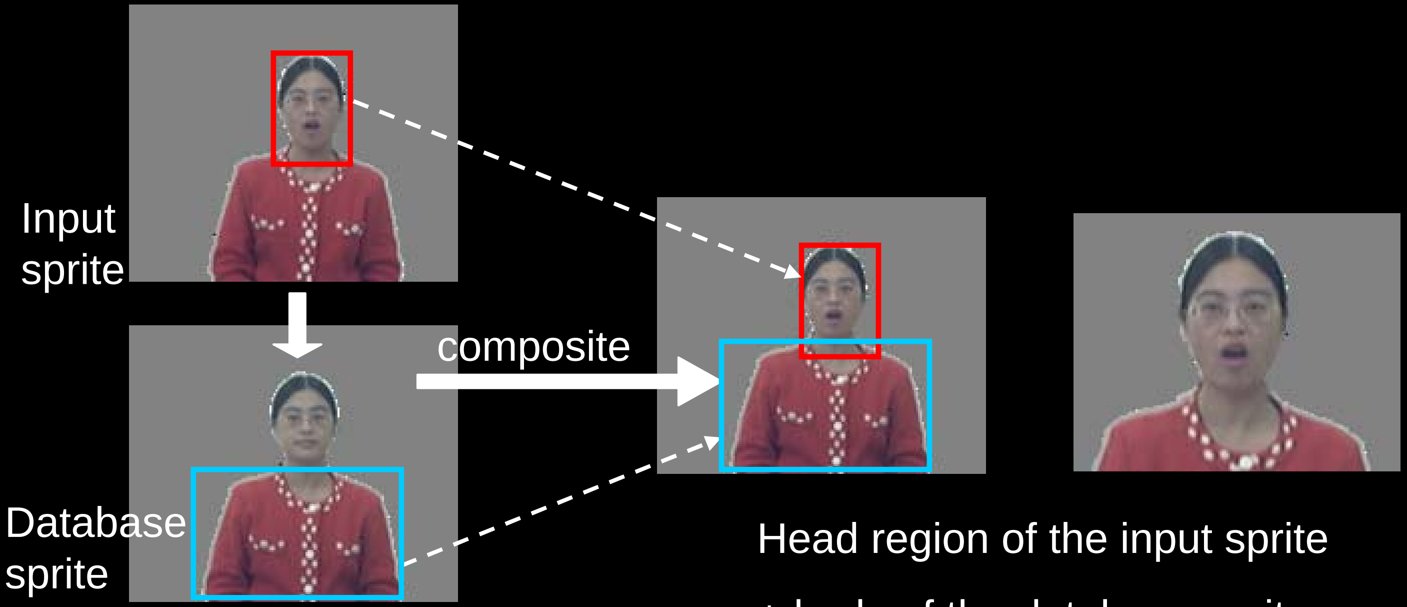
Sprites Comparison in Linear Searching



- Align the two sprites by 2D translation
- Comparison on shape and color using subset of pixels
- Sprites in the same group have higher searching priority (coherence)

Head Region Processing

- In high fidelity mode, head region of the input sprite is detected, compressed, sent to the rendering site, and composited with the matching database sprite

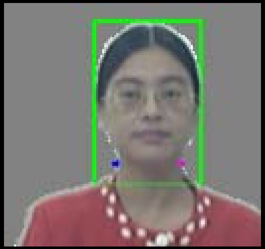


Head region of the input sprite

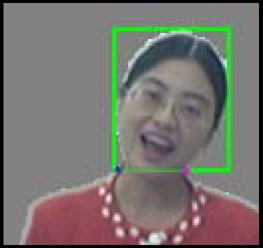
+ body of the database sprite

Head Region Detection and Composition

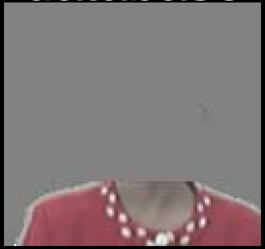
- Detection of bounding box of head region
- Composition
  - □ Neck alignment
  - □ Blending in transitional area below the head region



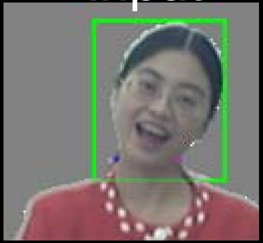
database



input



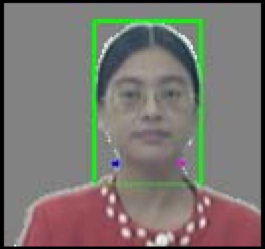
database

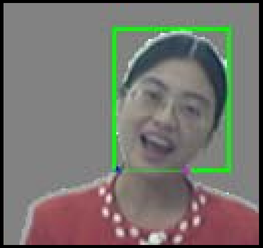


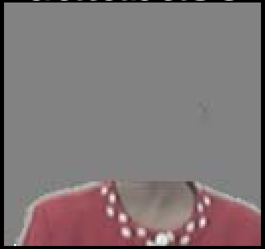
composite

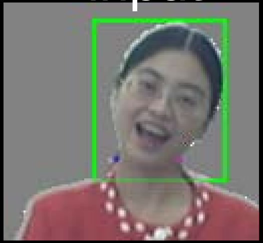
Head Region Detection and Composition

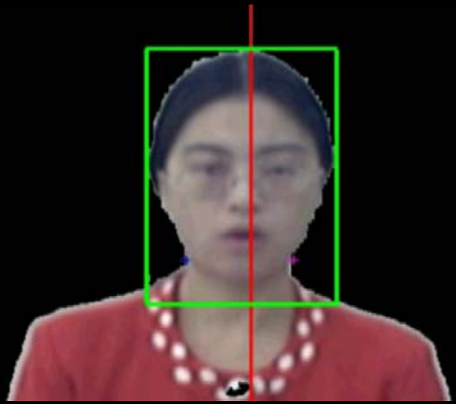
- Detection of bounding box of head region
- Composition
  - □ Neck alignment
  - □ Blending in transitional area below the head region
  - □ Using k-most recent frames to estimate the frame-to-frame changes of the position of the head.

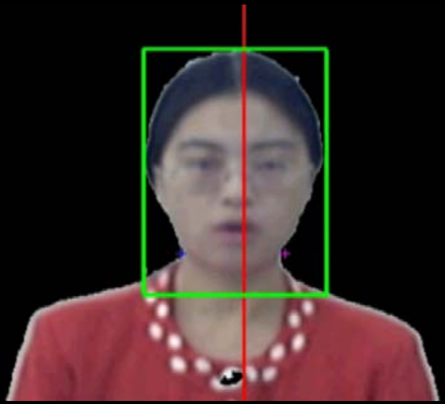

database


input

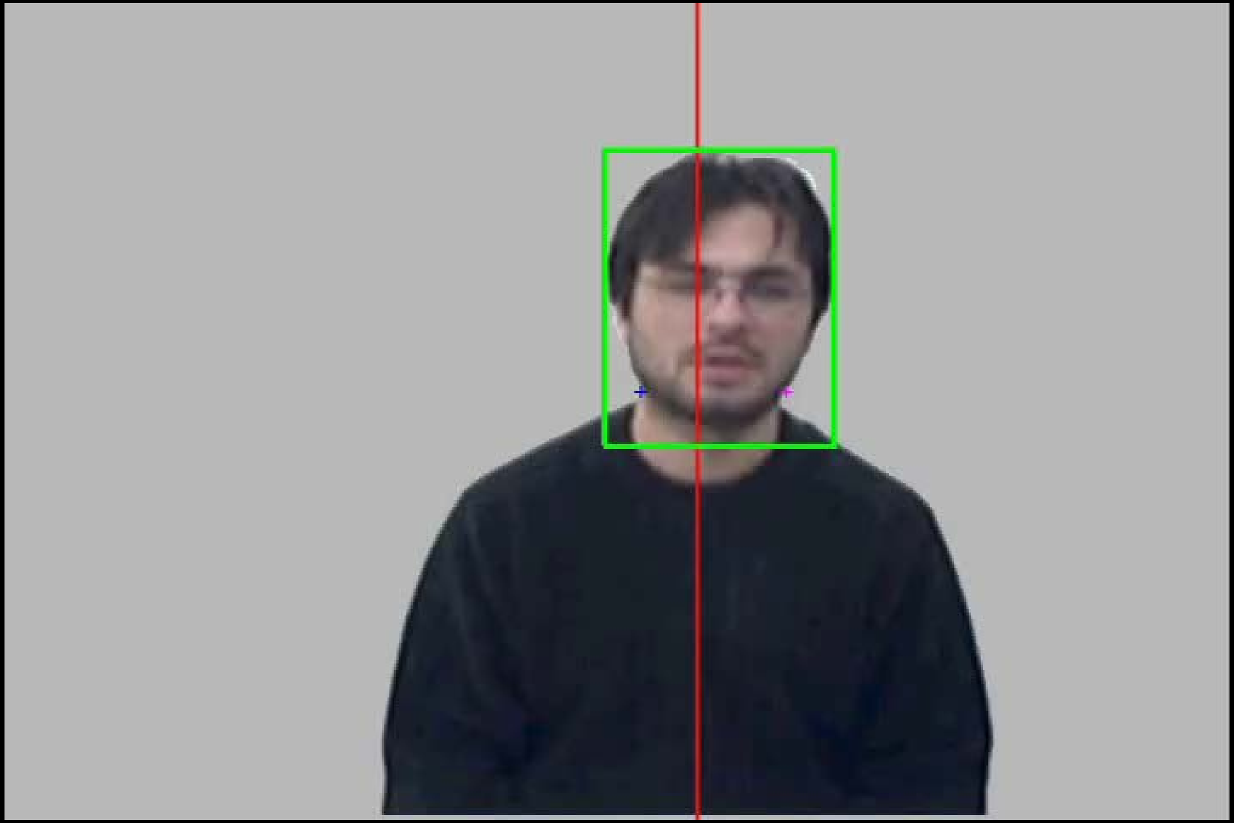

database


composite


K = 1


K = 25

Head Region Detection and Composition



Results

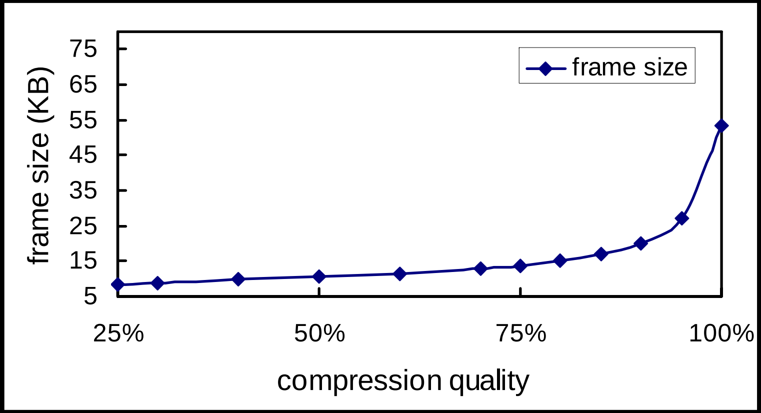
- Test on four different subjects
- 640x480 Video raw footage frames at 30fps

Performance at acquisition site

- Searching time on 1000 downsampled 160x120 frames (100 MB)
  - Brute force linear searching: 500 ms
  - Binary tree searching all sprites at the current leaf: 250 ms
  - Use coherence: 120ms
- Database uploading time for 2000 compressed sprites (95% quality factor)
  - 360 kbps uploading speed
  - 20/10/5 minutes for 640x480/320x240/160x120 resolutions

Performance at rendering site

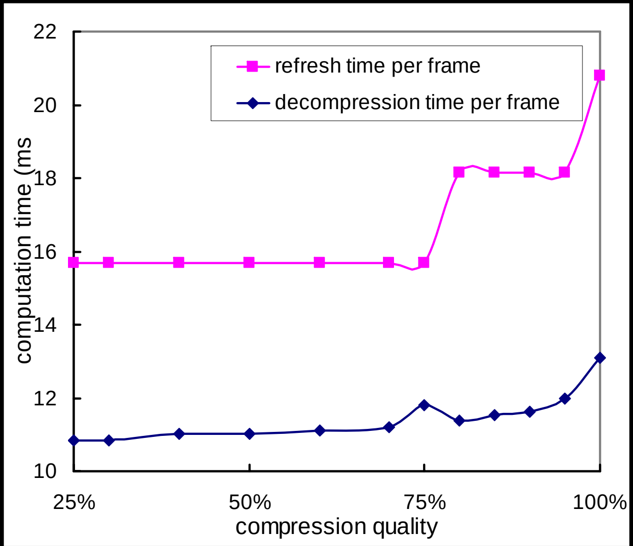
- Size of the stored database depends on the resolution of the frame, compression quality factor and the complexity of the texture
- The renderer can easily handle 20 databases of 2000 640x480 sprites (1.2 GB)



Memory size of a 640x480 sprite as a function of the compression quality factor

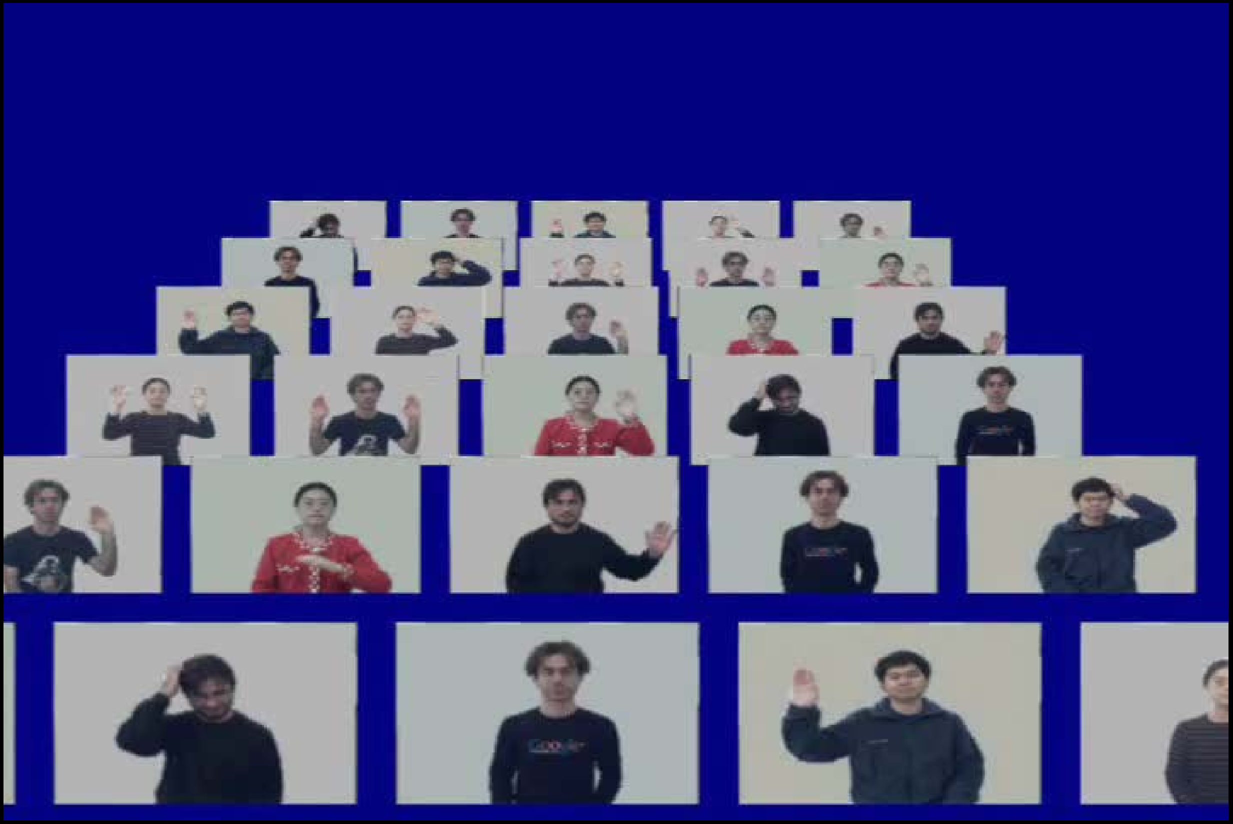
Performance at rendering site

- The decompression time depends little on compression quality factor
- Decompression time determines the number of databases to be loaded
- Average decompression time for 95% quality factor
  - 12/3.5/1.0 ms for 640x480/320x240/160x120 resolutions



Total frame time and decompression time of 640x480 sprite as a function of the compression quality factor

Simulated Renderer



Simulated renderer with 30 databases of 2000 160x120 sprites (540 MB) at 25fps

Simulated Renderer

Simulated renderer with 6 databases of 2000 640x480 sprites (360 MB) at 10 fps

Conclusion

- Our method
  - Models seated human in real time for teleconferencing and distance learning application
  - Drastically reduces the transmission data by sending 16 bits index per sprite
  - Makes best use of MTU if sending the index with the audio data

Future Work

- Apply our method in the context of an actual distance learning system
- Use as analytical tool for video archive
  - Video summarization
  - Video abstraction

Acknowledgement

- Radu Dondera, Mihai Mudure and Yi Xu
- Cristina Nita-Rotaru
- NSF grant SCI-0417458