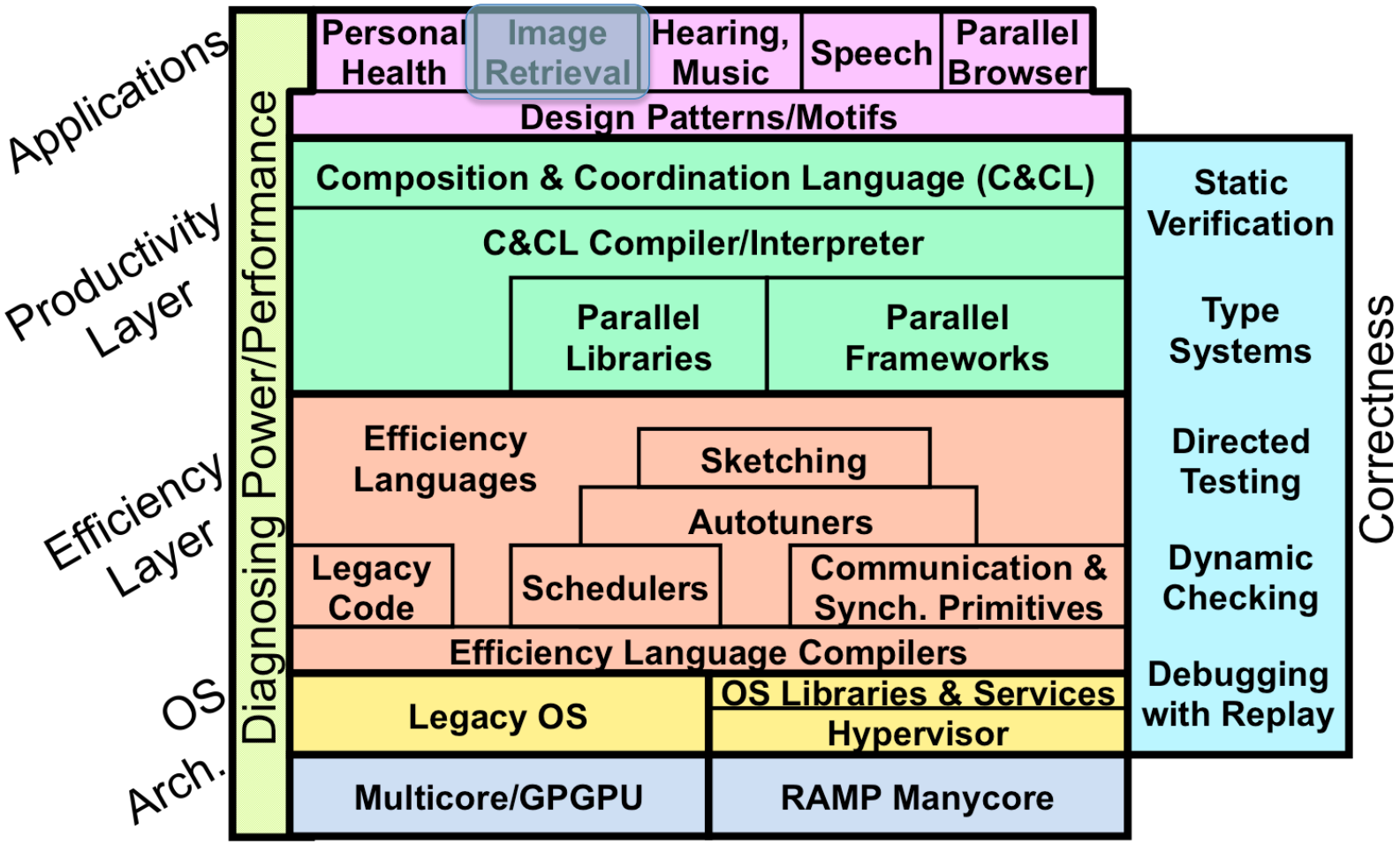
Video Segmentation

Narayanan Sundaram

Kurt Keutzer

Par Lab retreat, June 2011

Parlab Stack



Video Segmentation

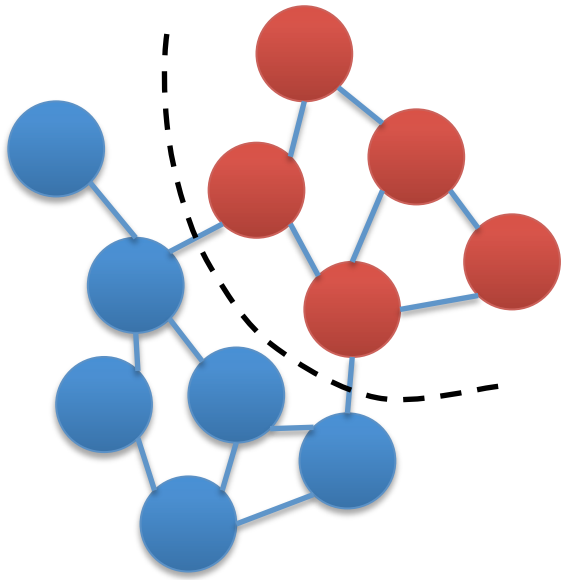
- Segment objects from a video sequence based on appearance and motion
- Required for video editing and video understanding
  - Video editing (object copy/ paste etc.)
  - Structure from motion (2D to 3D conversion etc.)
  - Scene context analysis
  - Understanding dynamic object interaction
- Not just parallelizing existing algorithms, but developing new algorithms

3

Computational requirements for video

- Video processing has traditionally been difficult because of data explosion
  - 1080p frames at 30 fps = 11 GB of uncompressed data per minute!
- Highly accurate computer vision techniques for images are hard to transition to video because of computational needs
- Video segmentation is highly compute intensive
  - We solve an eigenproblem defined over all the pixels in the video (>20 million pixels)

Introduction to Normalized Cuts



$$ncut(A,B) = \frac{w(A,B)}{w(A,V)} + \frac{w(A,B)}{w(B,V)}$$

$$A \cup B = V$$

- Minimizing normalized cut in a graph leads to an eigen problem
  - Find the generalized eigenvector corresponding to the smallest non-zero eigenvalue of $(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{y} = \lambda \boldsymbol{D}\boldsymbol{y}$
  - **W** is the affinity matrix and **D** = diag(**W**.**1**)

Video & Image Segmentation

- Our video segmentation algorithm is inspired by the gPb (global probability of boundary) image contour detection algorithm[1], which in turn is derived from Normalized Cuts

- gPb performs image contour detection by combining
  - Multiscale Pb (local boundaries)
  - Spectral Pb (normalized cuts)

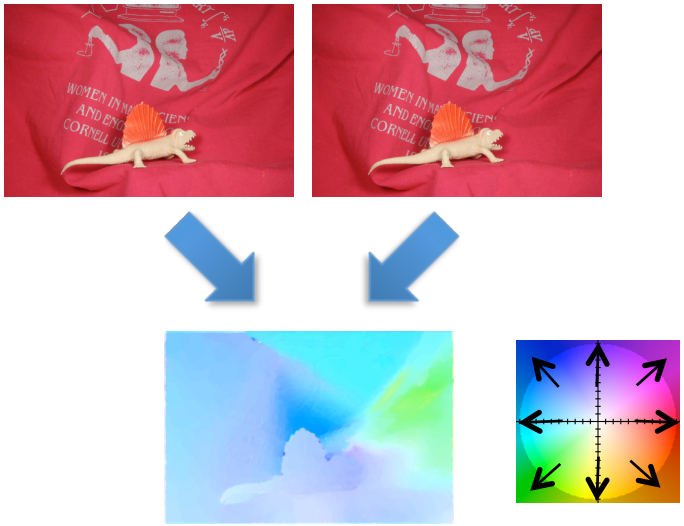
- Our previous work had reduced the runtime for this algorithm from 4 minutes to 2 seconds[2] per image.

[1] P. Arbelaez, M. Maire, and J. Malik. From contours to regions: An empirical evaluation. In CVPR, 2009.

[2] B. Catanzaro, B-Y. Su, N. Sundaram, Y. Lee, M. Murphy and K. Keutzer. Efficient, High Quality Image Contour Detector, In ICCV, September 2009

Motion Analysis

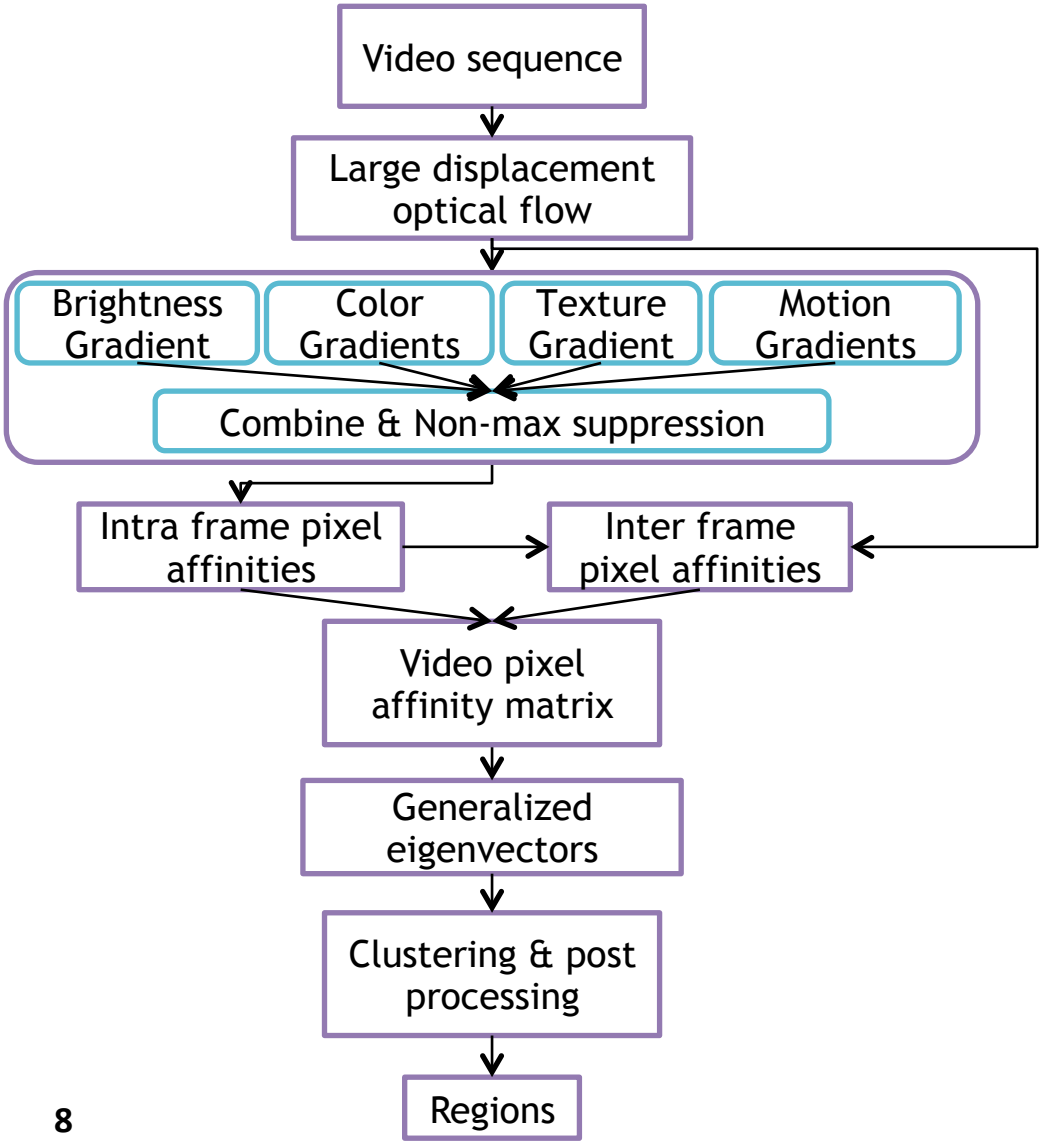
- Optical Flow is key to understanding motion in video
  - Optical Flow involves computing the motion vectors ("flow field") between the consecutive frames of a video
- We use the Large Displacement Optical Flow (LDOF) algorithm
  - Crucial for accurately measuring large motion of small objects.
- Our earlier work had reduced the time for this computation from 2 minutes to 4 seconds[3]

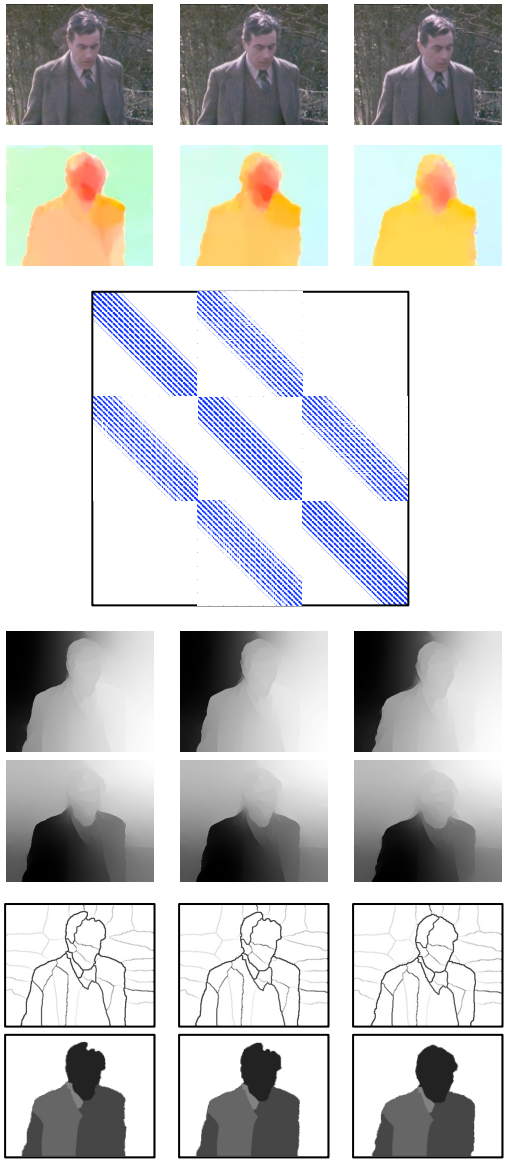


Hue indicates the direction of flow and saturation indicates the magnitude

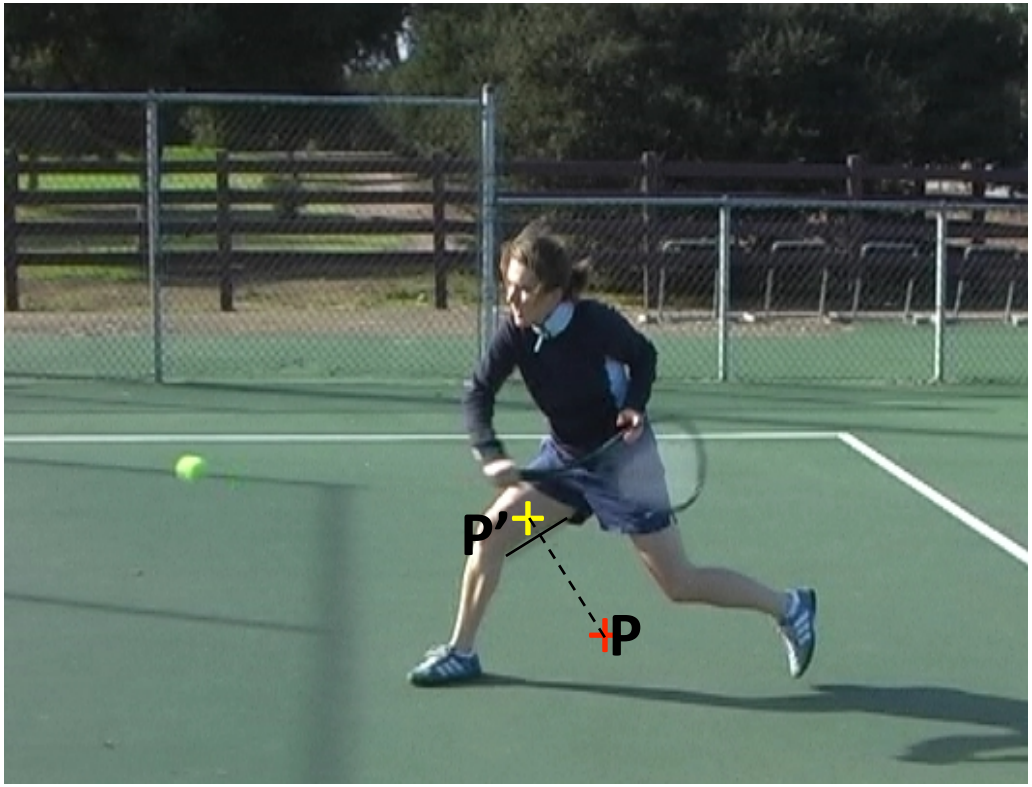
[3] N. Sundaram, T. Brox, K. Keutzer. Dense Point Trajectories by GPU-accelerated Large Displacement Optical Flow. In European Conference on Computer Vision (ECCV), September 2010

Our approach to video segmentation





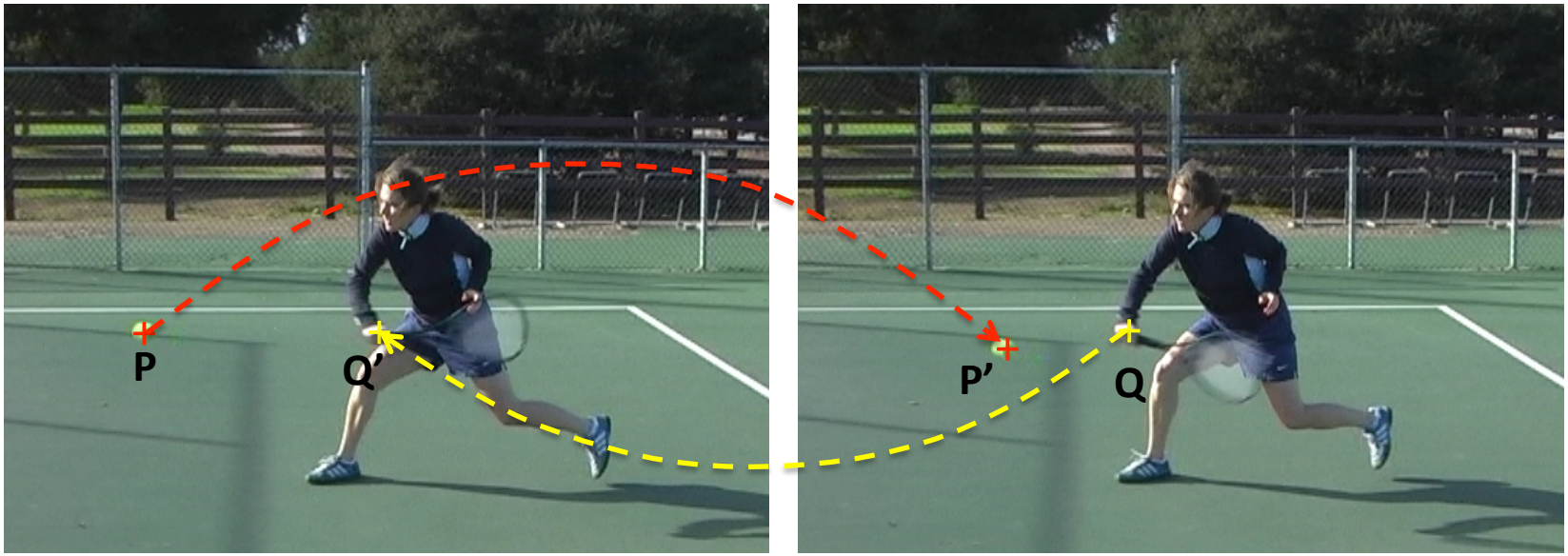
Computing affinities between pixels



- Affinities are computed within a frame using intervening contour
  - If there is a local edge in the line joining P and P', the affinity is low and vice-versa

From intra-frame to inter-frame pixel affinities

- Pixel affinity calculation between frames must be symmetric and must track motion
- We use Large displacement optical flow for motion tracking



Correspondence through Forward optical flow

Correspondence through Backward optical flow

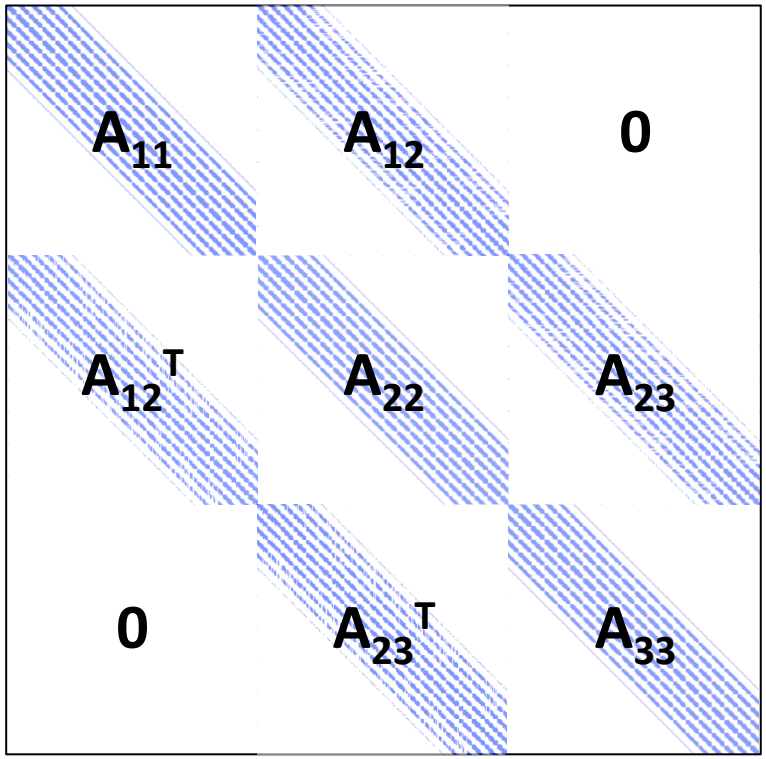
$$f(P,Q) = \min(f(P',Q), f(P,Q'))$$

Computations in eigensolver

- Need to perform a generalized eigensolve on the pixel affinity matrix ($Ax = \lambda x$)
- We need the eigenvectors corresponding to the *n* smallest eigenvalues
  - Lanczos algorithm
- Matrix is very large
  - ~20 GB for 100 frames of size 640x480
- We use a GPU+multicore CPU based cluster at NERSC for solving our problem
- We parallelized the computation across multiple levels (cluster, GPU, multicore CPU) with MPI, CUDA, OpenMP
  - Computation would be infeasible without parallelization

Challenges

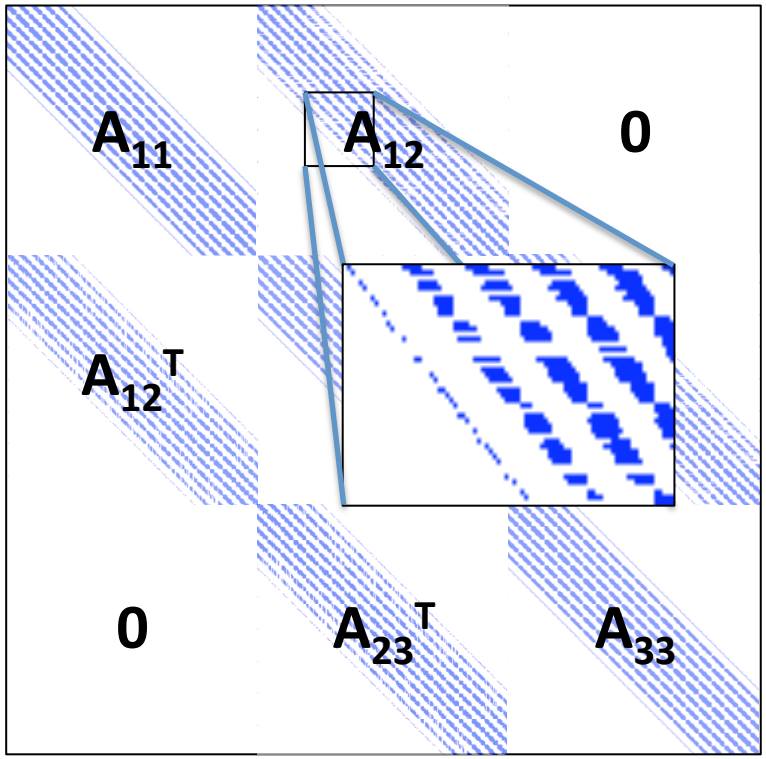
- Computing $A_{ij}^T x$ is slow as A_{ij} is not symmetric (CSR vs CSC)
  - A_{ij} has a special representation (same number & pattern of non-zeros/row)
- Non-determinism with floating point arithmetic has led to bugs
  - If we run out of GPU memory, the Lanczos algorithm is run twice
  - Divergence in these steps can lead to non-convergence



Affinity matrix structure for a sequence with 3 frames

Challenges

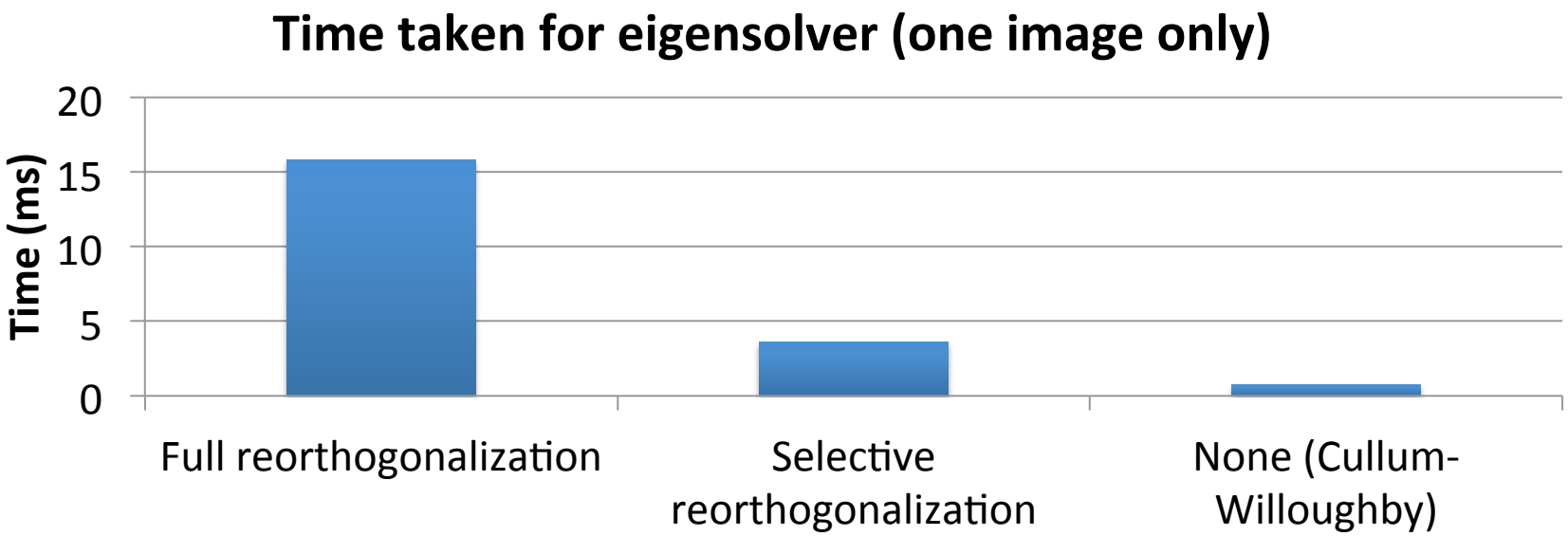
- Computing $A_{ij}^T x$ is slow as A_{ij} is not symmetric (CSR vs CSC)
  - A_{ij} has a special representation (same number & pattern of non-zeros/row)
- Non-determinism with floating point arithmetic has led to bugs
  - If we run out of GPU memory, the Lanczos algorithm is run twice
  - Divergence in these steps can lead to non-convergence



Affinity matrix structure for a sequence with 3 frames

Algorithmic Exploration

- Even with the Lanczos algorithm, several variations are possible for eigenvector reorthogonalization
- We use no reorthogonalization
  - Possible because our problem has well separated eigenvalues
  - Fixed at the end using the Cullum-Willoughby test
- 20X performance improvement from this algorithm (compared to default full reorthogonalization)

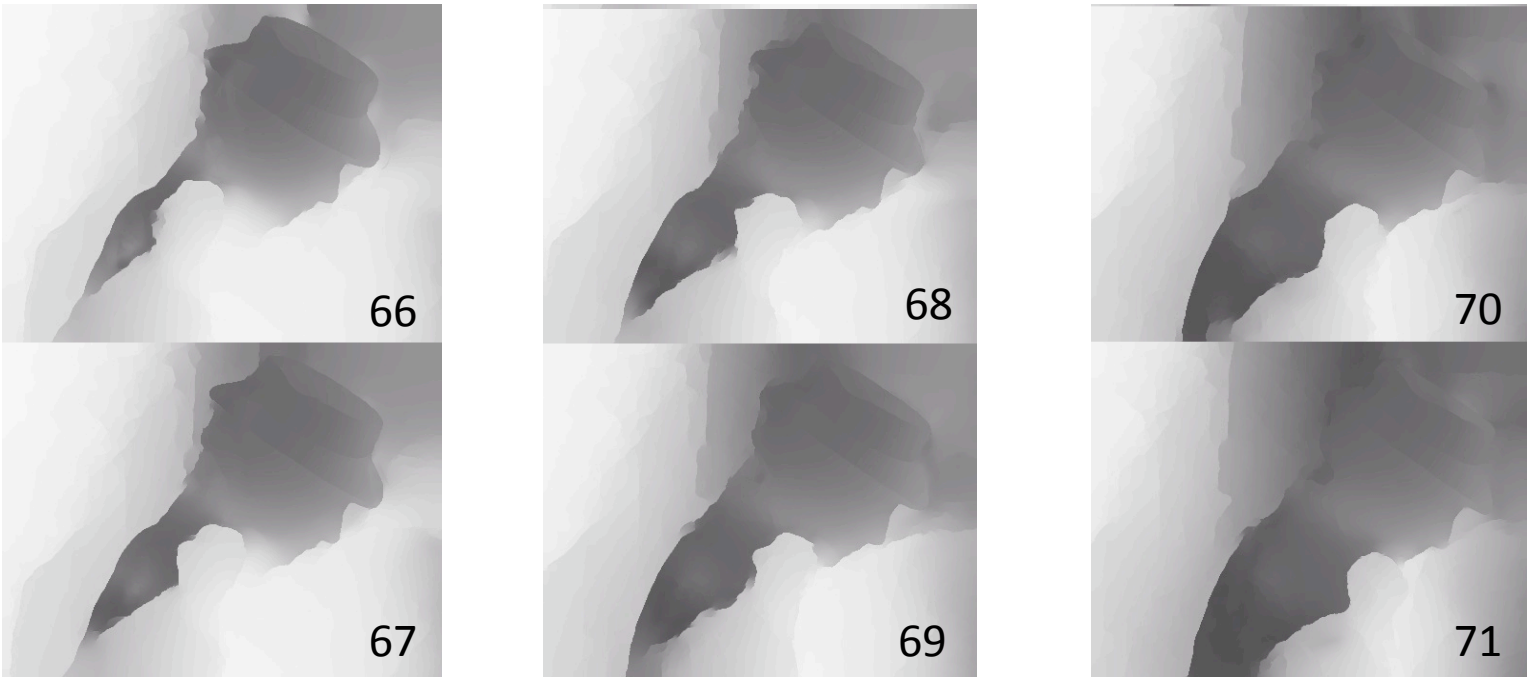


Leakage in eigenvectors

We cannot use the eigenvectors without postprocessing.
“Leakage” is caused by smooth transitions.

15

Smooth transitions

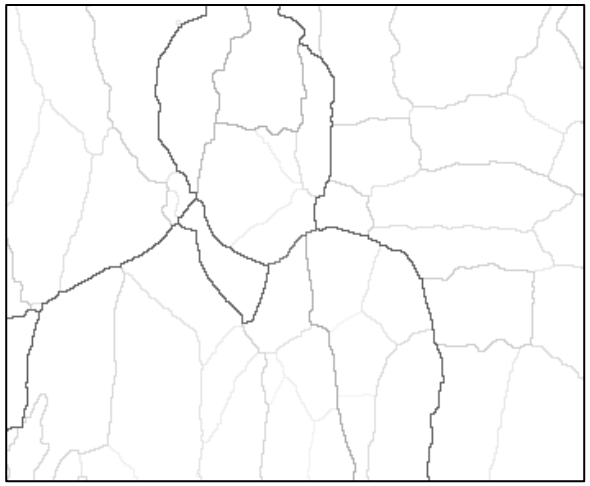
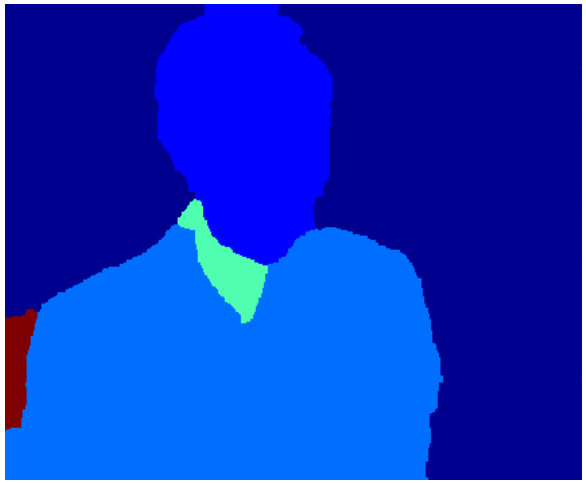


Use a clustering approach to group pixels locally and then combine the groups globally

16

Clustering & weighting

- Perform k-means clustering on the eigenvectors to get an oversegmentation
  - We use k = 500
- Use Ultrametric Contour Maps to join regions based on the strength of the boundary between them
  - Gives a hierarchical segmentation

Video Segmentation - Results

Algorithm	Label density	Overall accuracy (% pixels correctly labeled)	Objects extracted with <10% error	Over-segmentation
Brox & Malik - trajectory segmentation [4]	3.43%	3.17%	**7**	**3.14**
ALC with incomplete tracks [5]	3.43%	2.77%	0	54.57
Hierarchical Graph Segmentation [6]	**100%**	79.23%	0	10.42
Our technique	**100%**	**84.54%**	3	5.86

[4] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In ECCV, 2010.
[5] S. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete or corrupted trajectories. In CVPR, 2008.
[6] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. CVPR, June 2010.

Videos

19

Video Segmentation - Results

- Compared to sparse motion segmentation techniques, we label 30x more points with better accuracy
- Compared to dense video segmentation techniques, we achieve better accuracy with 2x less oversegmentation
- Our implementation takes ~ 5 min for 200 frames with 34 nodes
  - Running this application on a similar sized cluster would have taken > 150 min with existing serial implementations

Summary

- We have designed & implemented a superior video segmentation algorithm that is denser and more accurate than existing algorithms.
- Parallelization has enabled us to move highly accurate computer vision algorithms from images to videos
  - We are using a cluster today, but expect it to run on the desktop within 5-10 years
- Efficient parallelization has enabled us to apply our algorithms at much larger scales (optical flow on HD video, video segmentation on 100's of frames)

THANK YOU

QUESTIONS?

References

[1] P. Arbelaez, M. Maire, and J. Malik. From contours to regions: An empirical evaluation. InCVPR,2009.

[2] B. Catanzaro, B-Y. Su, N. Sundaram, Y. Lee, M. Murphy and K. Keutzer. Efficient, High Quality Image Contour Detector, In ICCV, September 2009

[3] N. Sundaram, T. Brox, K. Keutzer. Dense Point Trajectories by GPU-accelerated Large Displacement Optical Flow. In European Conference on Computer Vision (ECCV), September 2010

[4] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In ECCV, 2010.

[5] S. Rao, R. Tron, R. Vidal, and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete or corrupted trajectories. In CVPR, 2008.

[6] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. CVPR, June 2010.

[7] T. Brox, J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. In IEEE Transactions on Pattern Analysis and Machine Intelligence, to appear

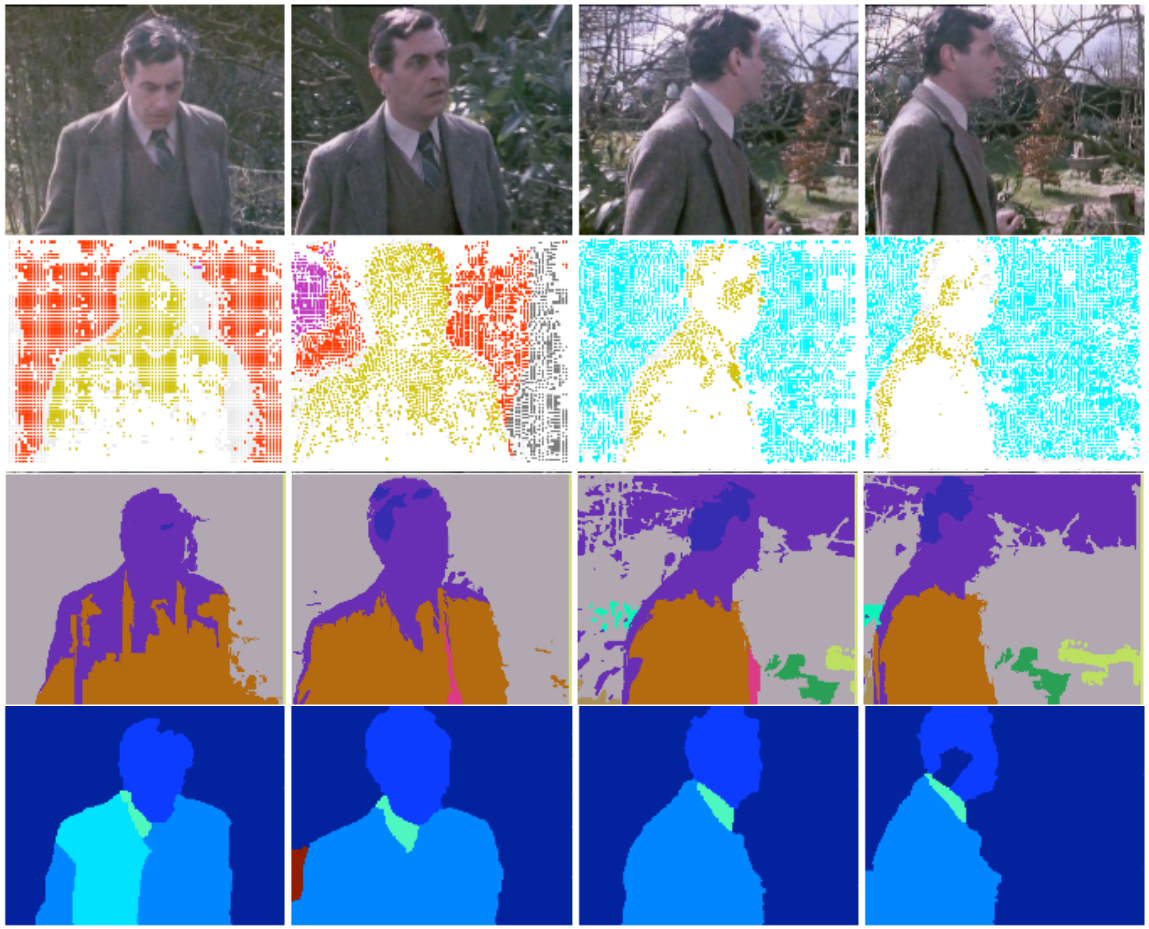
[8] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by GPU-accelerated large displacement optical flow. Technical Report UCB/EECS-2010-104, EECS Department, University of California, Berkeley, Jul 2010.

[9] Sand, P., Teller, S.: Particle video: Long-range motion estimation using point trajectories. International Journal of Computer Vision 80 (2008) 72–91

[10] Shi, J., Tomasi, C.: Good features to track. In: CVPR. (1994) 593–600

[11] Zach, C., Gallup, D., Frahm, J.M.: Fast gain-adaptive KLT tracking on the GPU. CVPR Workshop on Visual Computer Vision on GPU's (CVGPU) (2008)

Video Segmentation - Results



Original video

Motion segmentation
Brox & Malik [4]

Hierarchical graph
Segmentation

Our approach
Final segmentation