Utilizing GPUs for Rapid Facial Recognition using Video Input

Presented By: Mr. Charles Gala

Advisors:

Dr. Raj Acharya, Computer Science and Engineering
Mr. Bruce Einfalt, Applied Research Laboratory

Penn State University

Outline

- Background
- Recognition Method Structure
- Environment and Evaluation
- Results
- Conclusions

Background

- **Facial recognition** is an active research area that provides a real-time application of pattern recognition techniques

- Significant challenges to utilizing **live streaming video** for input
  - Processing speed of method vital for accessing frames

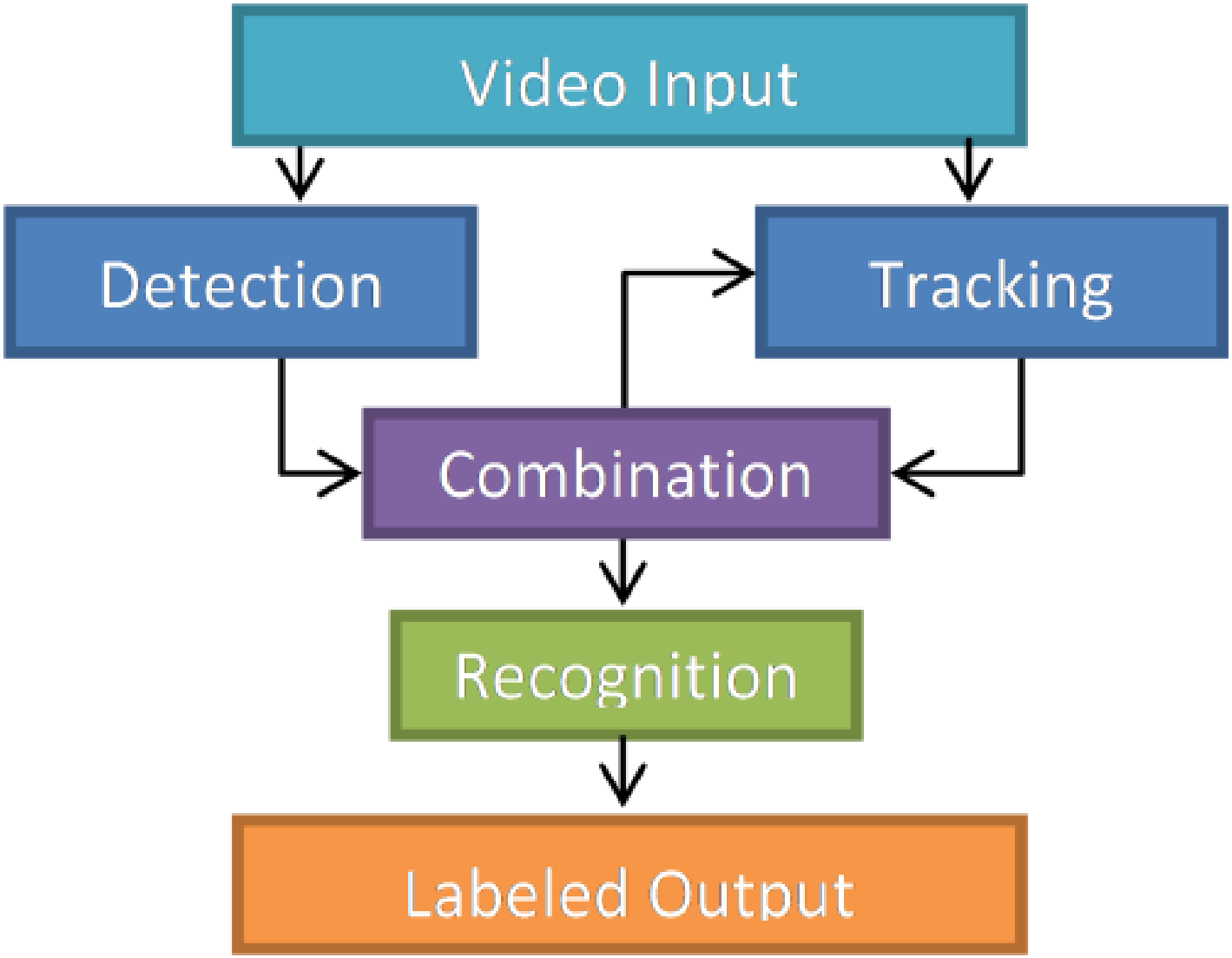
Background (Cont.)

- **Graphics Processing Units (GPUs)** are an ideal way to accelerate recognition processing
  - Support large number of cores for parallel processing

- Goal is to develop a recognition method that utilizes GPU parallelization

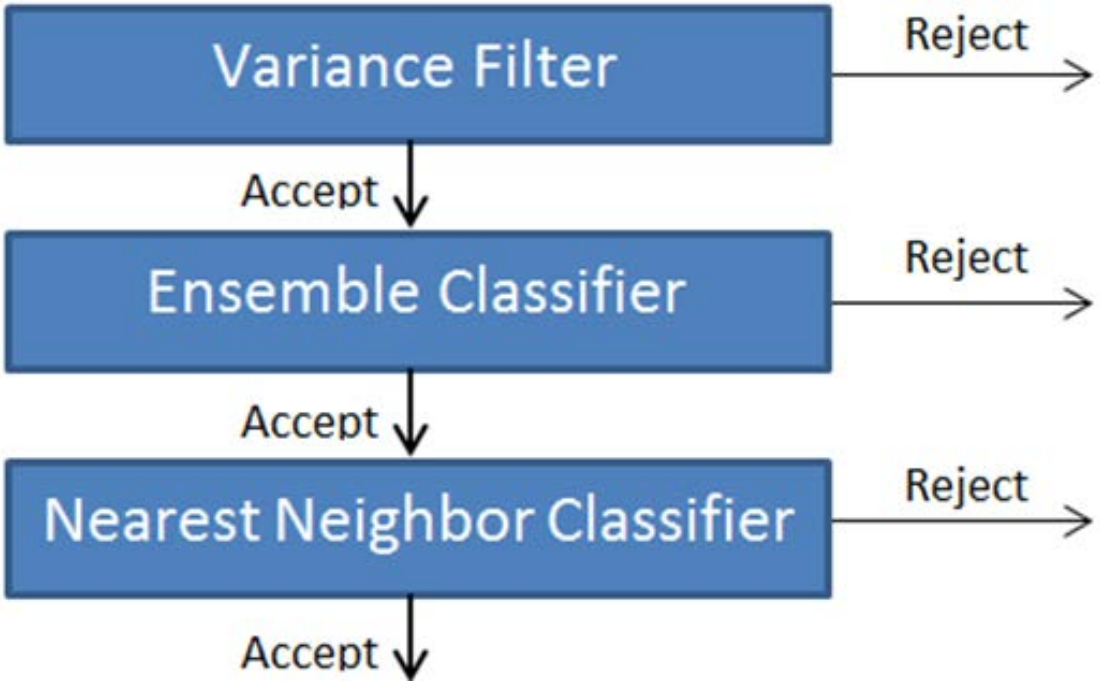
PENNSTATE ARL

Recognition Method Structure



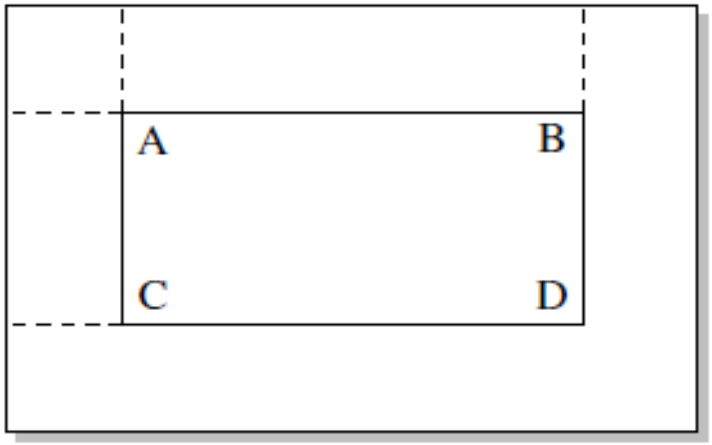
Detection

- **The Detection stage** finds new instances of some target pattern (eg a face)
  - Sliding window approach used to detect patterns



Variance Filter

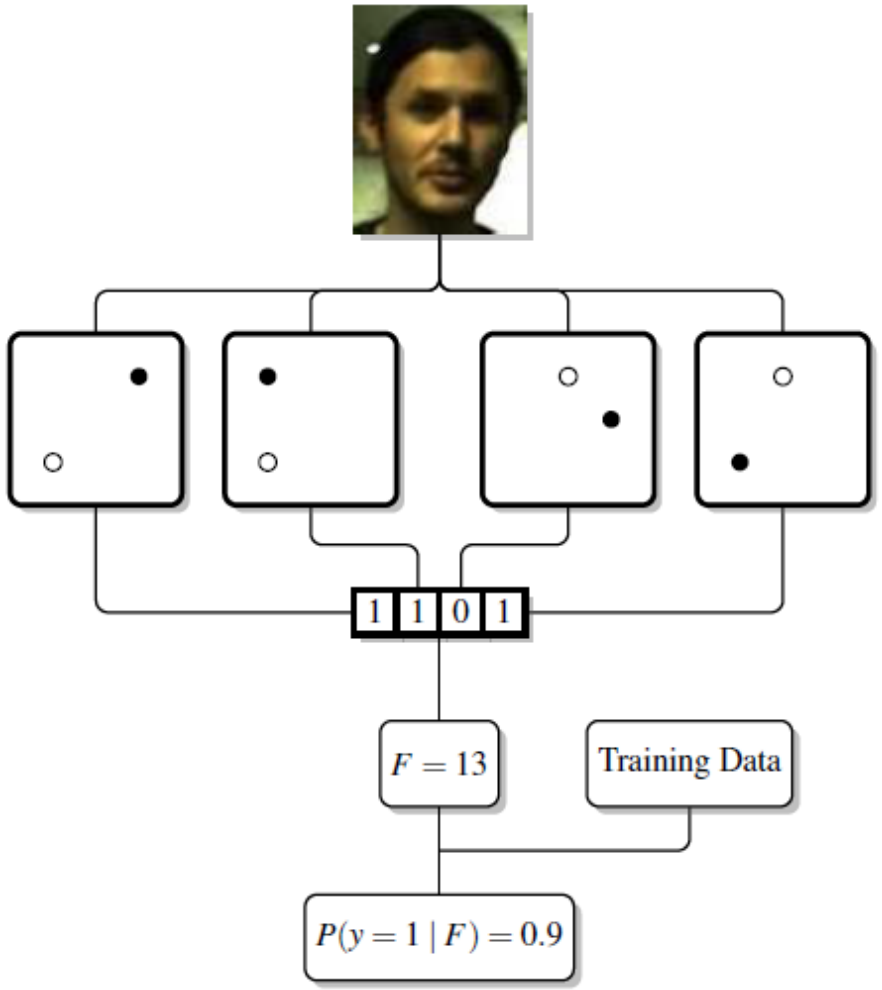
- Low variance windows have few meaningful features
  - Filter rejects such windows (< threshold)
- Integral images calculated to quickly find variance of each window
  - Four memory lookups to calculate sum of region



Sum of Region =
A - B - C + D

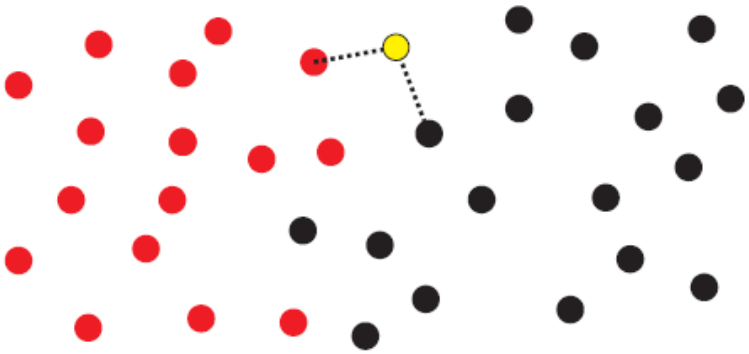
Ensemble Classifier

- Evaluates each window using a series of randomized ferns, takes average
- Each fern utilizes a set of 2-bit binary features
- Minimal number of memory lookups per fern



Nearest-Neighbor Classifier

- Evaluates using a stored set of positive and negative examples
- Finds the distance to nearest positive and negative example, then calculates a ratio for the confidence value
- Distance based on Norm Cross Correlation



$$confidence = \frac{d^-}{d^- + d^+}$$

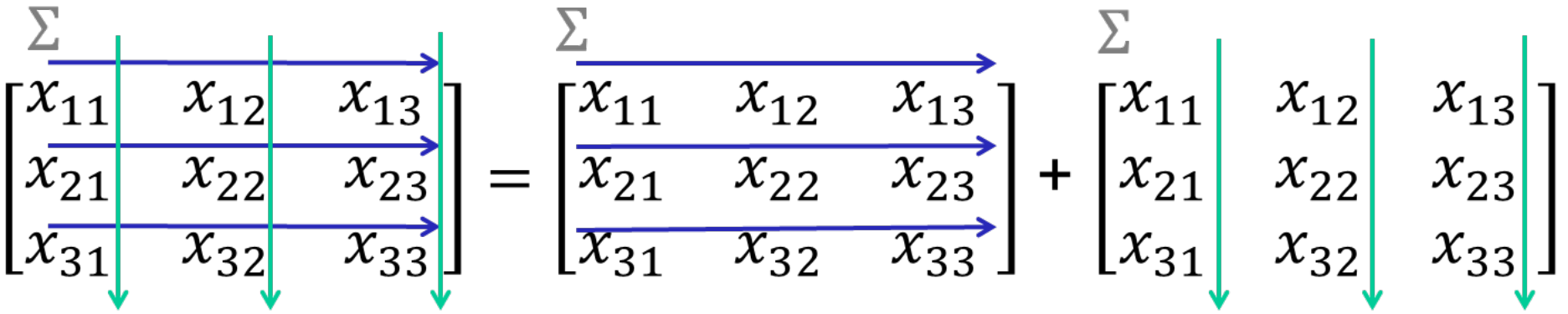
GPU Acceleration- Detection

- Split detection cascade into two segments
  1) Variance Filter/Ensemble Classifier
  2) Nearest Neighbor Classifier

- **Variance Filter/Ensemble Classifier**- Window evaluations are performed in parallel on the GPU
  - Separate GPU thread for each window
  - Results stored in an array that is copied back to the CPU upon completion

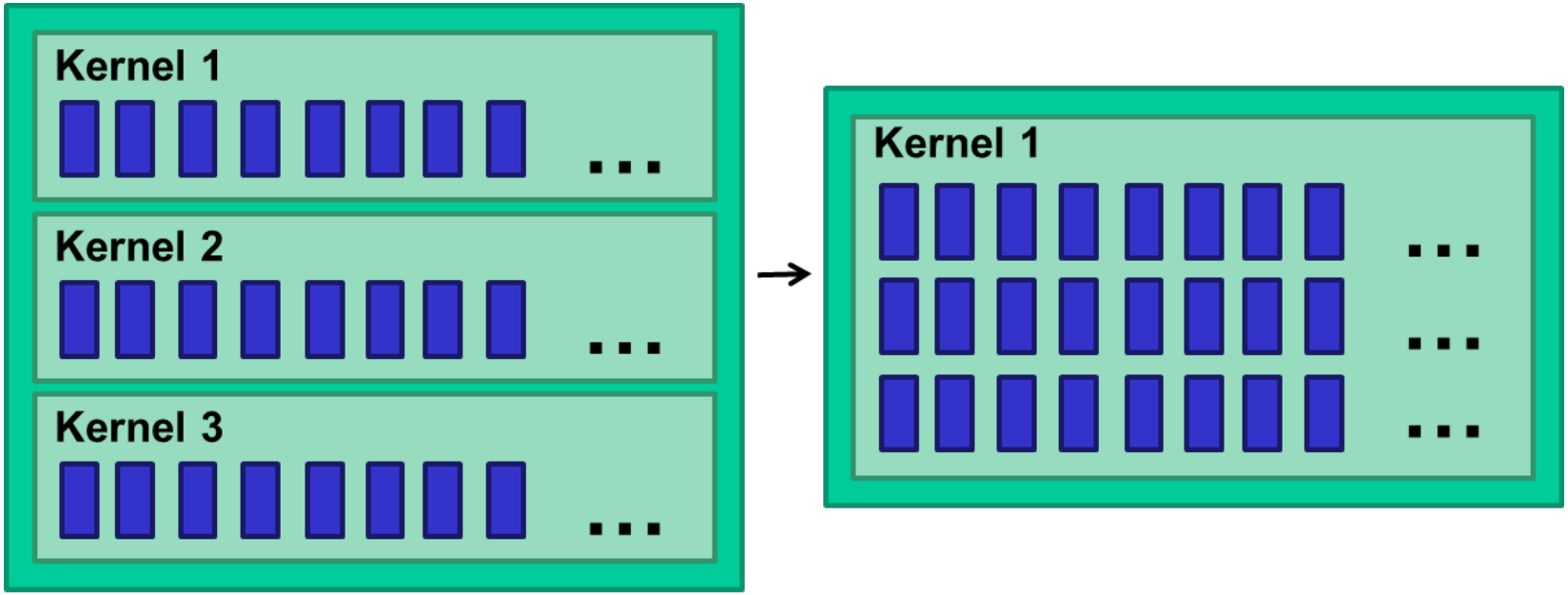
GPU Acceleration (Cont.)

- **Variance Filter**- New integral images must be found for each frame of the input sequence
- The integral image calculations can be modified to allow for parallelization
  - Introduced by Bilgic et al.



GPU Acceleration (Cont.)

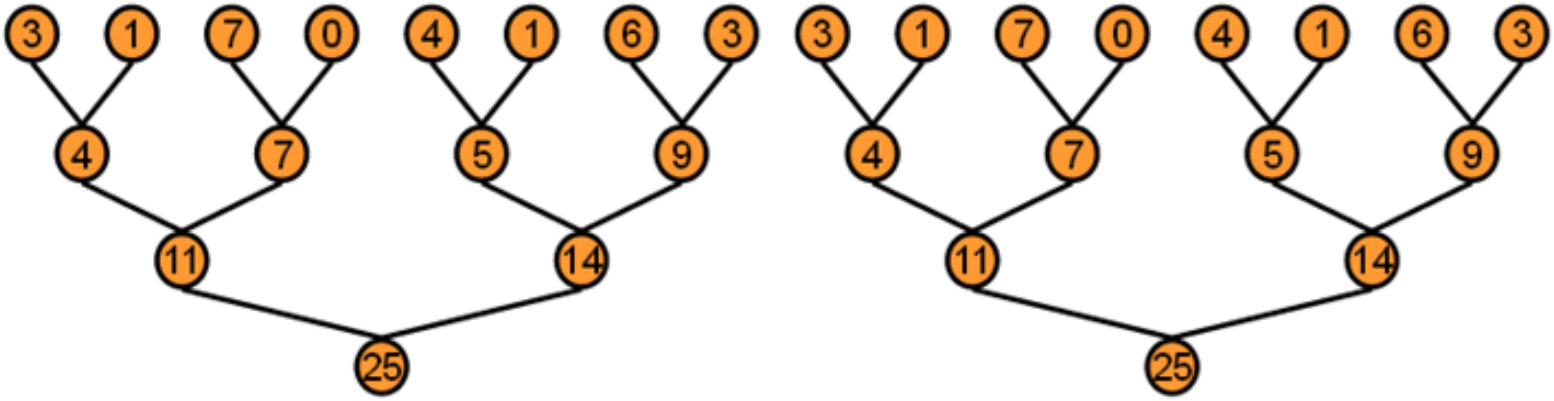
- **Nearest-Neighbor Classifier**- Cannot evaluate in parallel the same way as the previous classifiers
- More efficient to evaluate examples for each window in parallel instead



- Texture and shared memory used in implementation

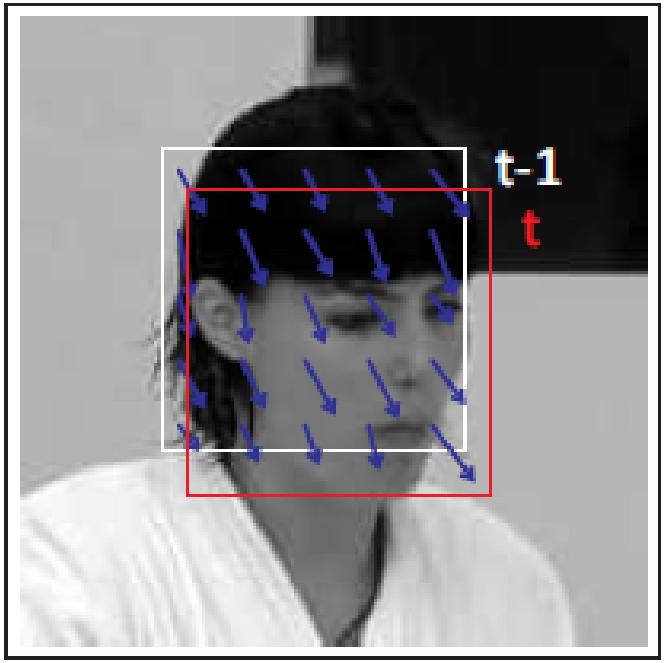
GPU Acceleration (Cont.)

- **Nearest-Neighbor Classifier**- Nearest positive and negative examples must be found for a given window
- Parallel reduction can be used here
  - Perform all reductions at the same time and is treated as one large reduction



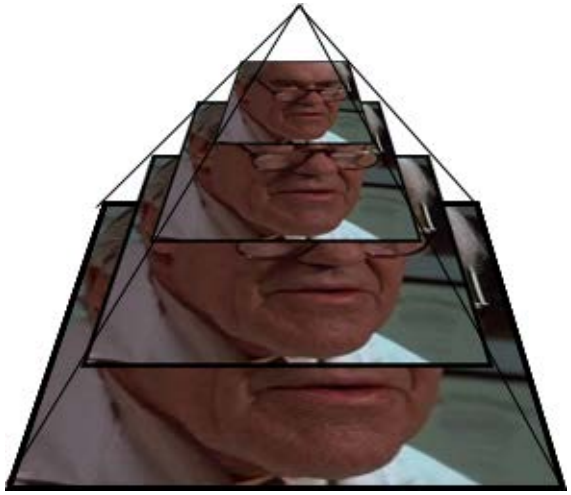
Tracking

- **The Tracking stage** estimates the movement of previously found patterns between frames
- Generates a set of 400+ data points to track
- Resulting motion estimates used to calculate the new bounding box



Tracking (Cont.)

- **Lucas-Kanade** tracking technique for optical flow estimation used (pyramidal implementation)
  - Obtain a rough estimate of the motion, then refine
  - Uses a collection of scaled images (Image Pyramid)



- Two error measures used to determine the reliability of the data point estimates

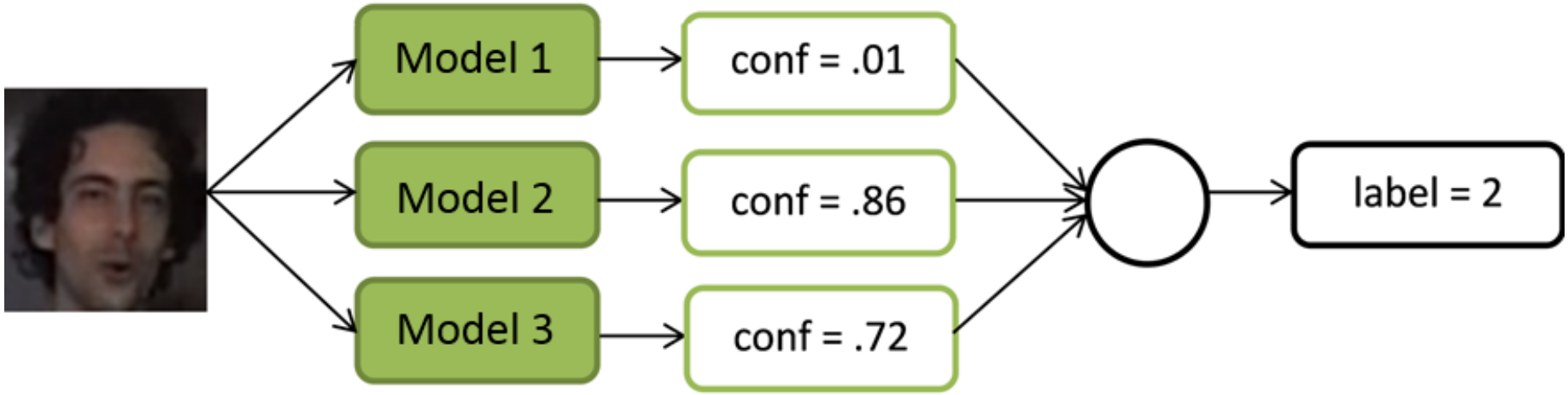
GPU Acceleration- Tracking

- Each data point is evaluated independently on a separate GPU thread
  - Both Lucas Kanade tracking and error calculations

- Separate kernel call for each level of the image pyramid (six calls total)
- Image pyramid information stored on texture memory in the GPU

Recognition

- **The Recognition stage** takes face images found from previous stages and identifies them
  - Distinguish each face from other faces that appear
- Applies a one-vs-all strategy to identify faces (model for each face)
  - No GPU acceleration used



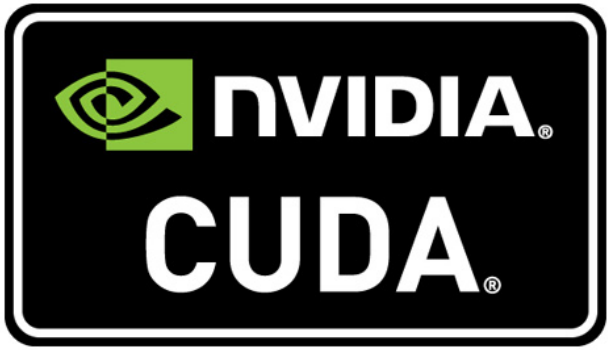
System Environment

- **CPU Implementation:**
  - All work left on the CPU, no memory copied to GPU
  - Intel Core 2 Duo Processor E6600 (2 Cores, 2.4 GHz per core)

- **GPU Implementation:**
  - GPU Accelerations used
  - NVIDIA Tesla C2050 (448 Cores, 1.15 GHz per core)

System Environment (Cont.)

- Both implementations written in C++
- OpenCV library used during operation
  - Provides CPU implementation of Lucas Kanade
  - Includes hardware accelerations
- CUDA and Thrust parallel algorithms library used for GPU implementation



Evaluation Criteria

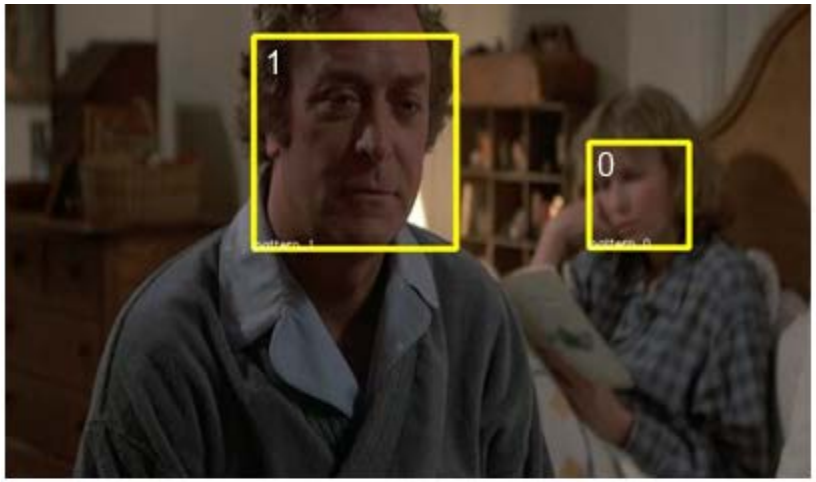
- Recognition method run in two different modes
- **Offline Mode**: All frames of the video sequence are available to the method
- **Live Mode**: Next available frame from video sequence depends on time take to process previous frame (simulated at 25 fps)

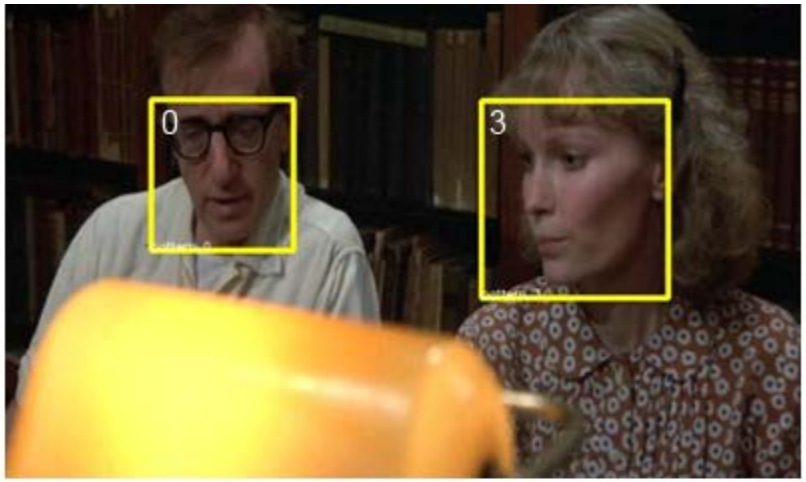
Datasets

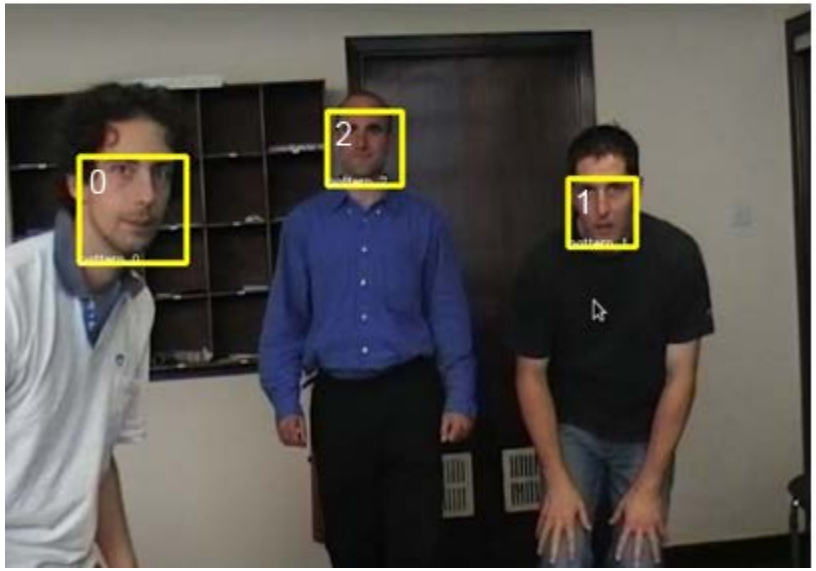
- **Hannah Dataset**- Manually annotated dataset for the 1986 film “Hannah and Her Sisters” by Woody Allen
- **SPEVI Dataset**- Contains sequences where several people that move around in a scene

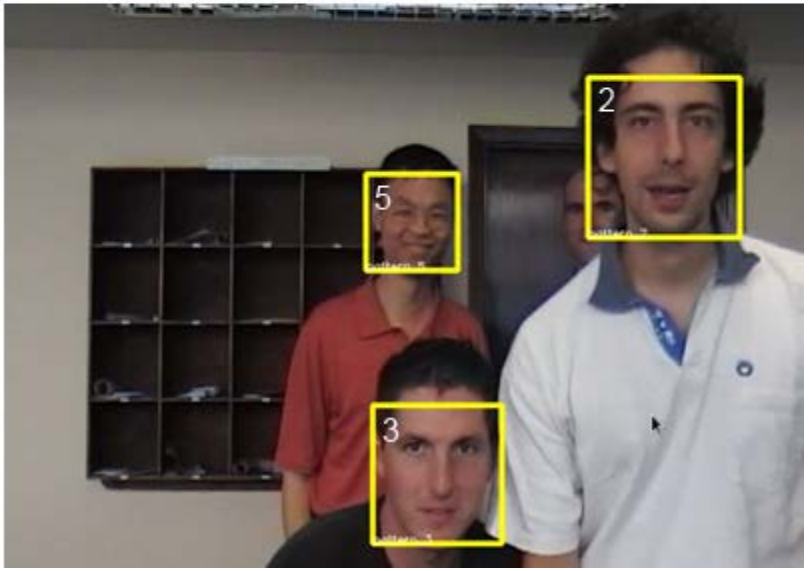
PENNSTATE ARL

Results - Overall









PENNSTATE ARL

Results - Overall (Cont.)

- **Offline Mode Evaluations**

	f-measure	recognition	fps
CPU	**0.874**	**0.821**	**10.31**
GPU	**0.880**	**0.841**	**35.72**

  - Difference in f-measure and recognition due to differences in hardware precision
  - GPU implementation had a speedup over the CPU implementation ranging from **2.5x to 6x**
  - Average speedup of **3.5x** overall

Results - Overall (Cont.)

- **Live Mode Evaluations**

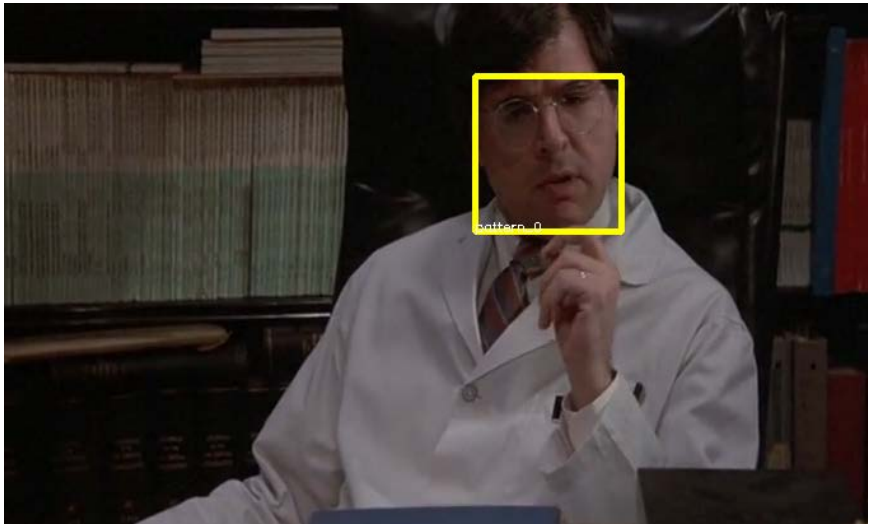
	f-measure	recognition	frames lost
CPU	0.716	0.769	2.29
GPU	0.830	0.794	0.75

– Percentage Loss (as compared to offline mode)

	f-measure	recognition
CPU	22%	7%
GPU	6%	6%

Conclusions

- Purpose of research was to develop a novel facial recognition method that utilizes GPU acceleration
- GPU acceleration is an excellent option to improve the processing runtime of an algorithm



Any Questions?

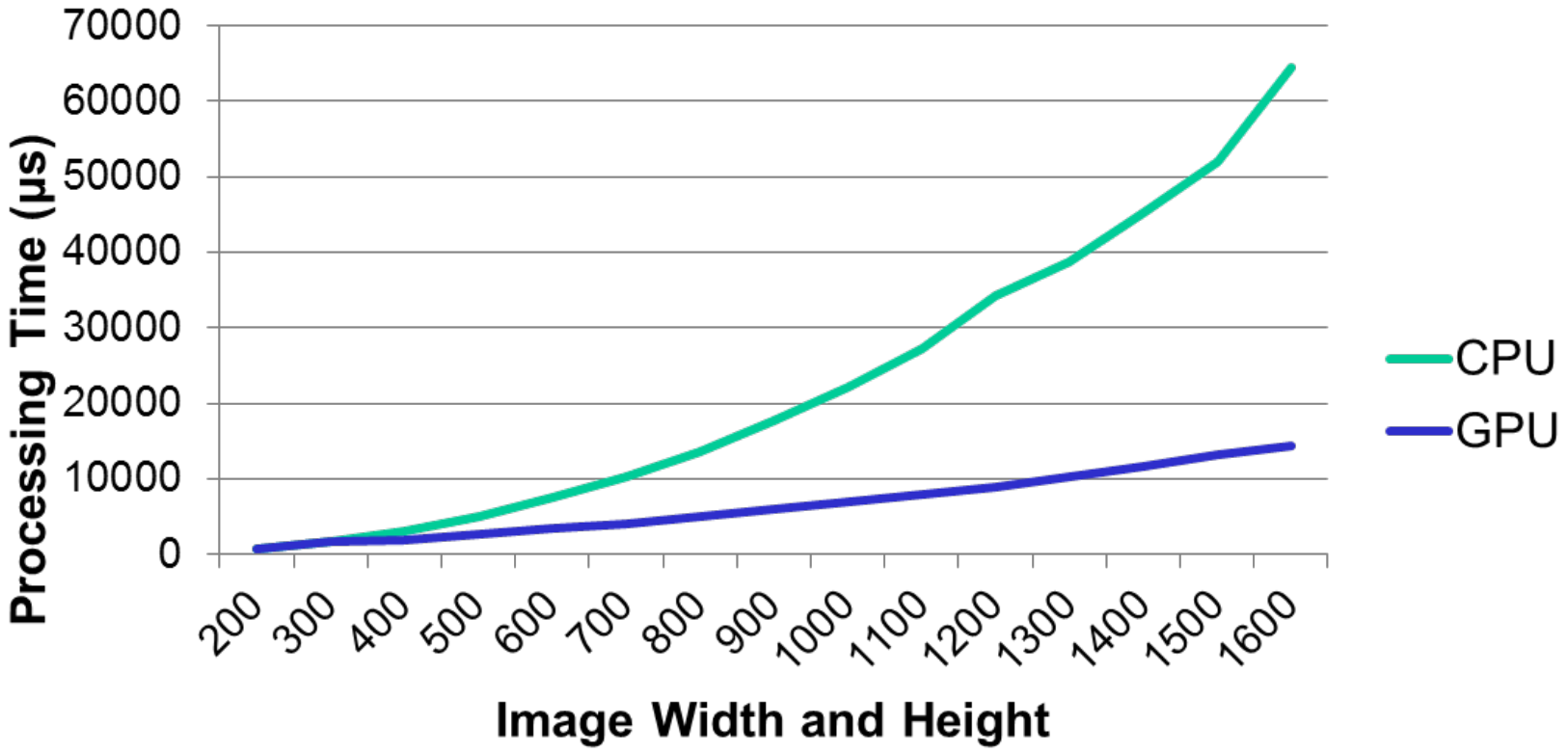
Thank you for your time!

Contacts

- **Contact information**
  - Mr. Charles Gala (cjgala@gmail.com)
- **Original research paper that this paper is based on:**
  - https://etda.libraries.psu.edu/paper/21868/

Results - Component

- **Integral Image Calculations**



At the largest image size the GPU calculations had a 4.5x speedup over the CPU calculations.

Results - Component (Cont.)

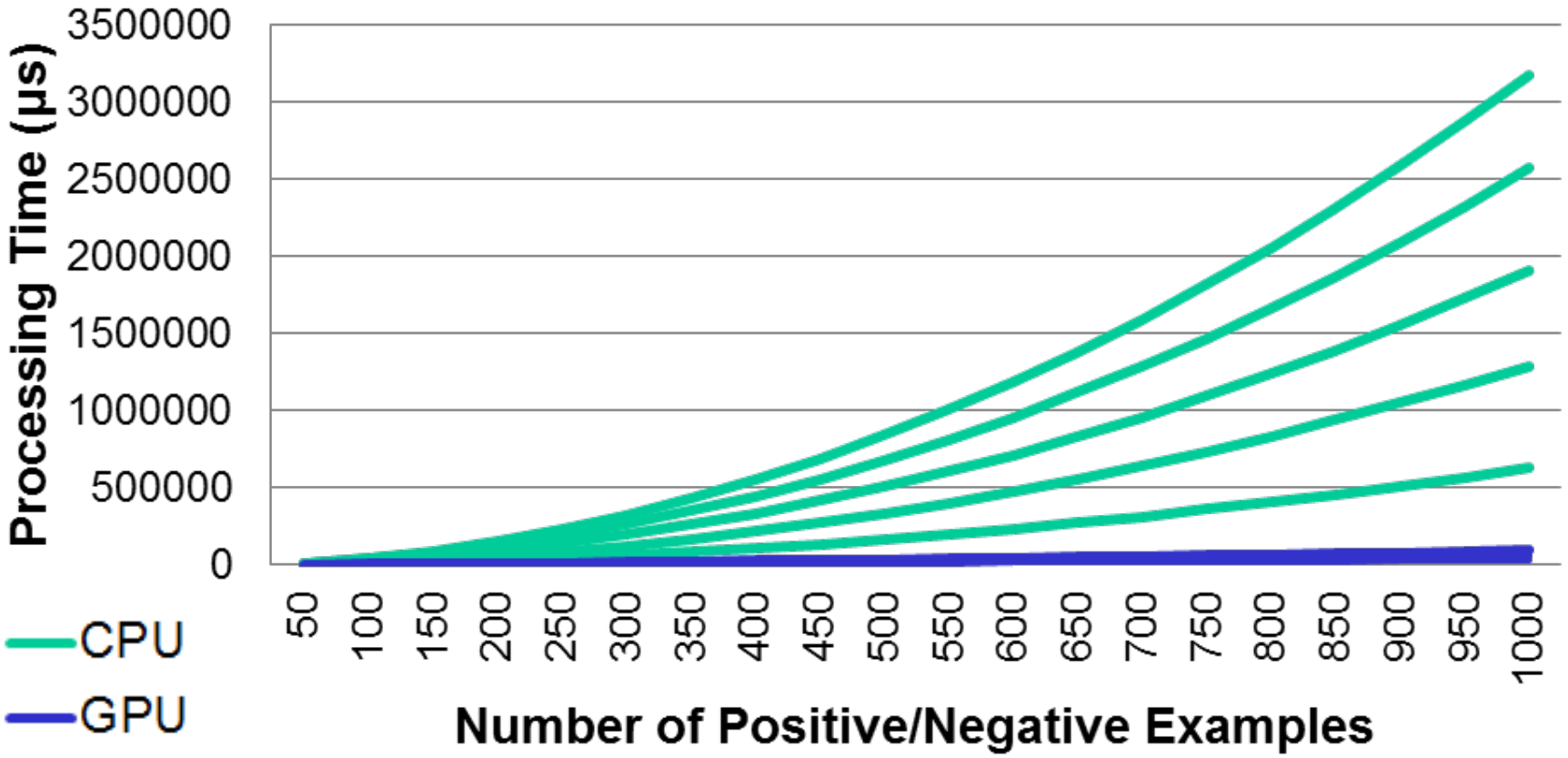
- **Variance Filter / Ensemble Classifier Evaluations**
  - Average processing time to evaluate all eight sequences

Average CPU Runtime: 10350 µs
Average GPU Runtime: 1230 µs

  - Speedup gained in using the GPU accelerations ranges from **7x to 10x**, with an average speedup of **8.39x**

Results - Component (Cont.)

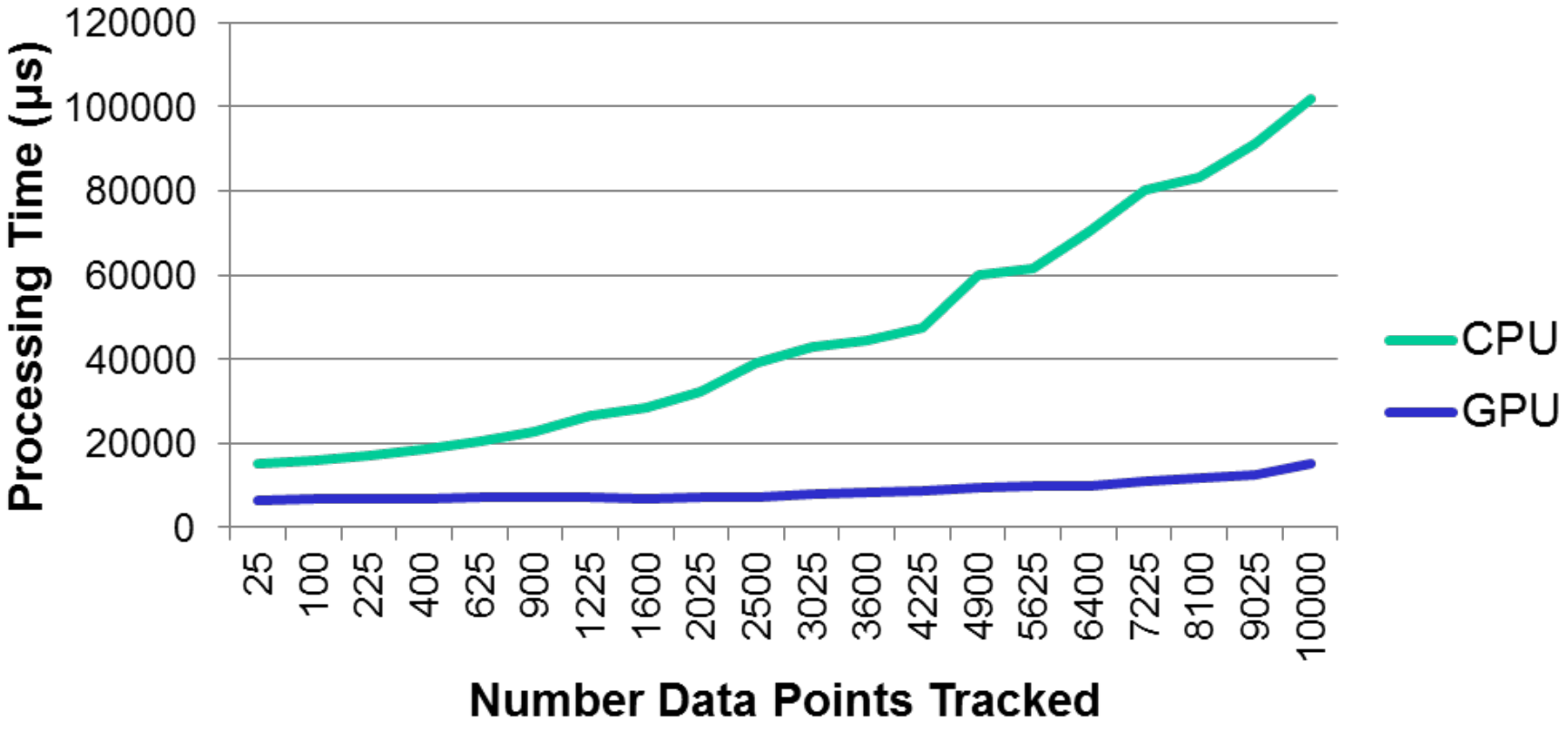
- **Nearest-Neighbor Classifier Evaluations**



The speedup achieved by the GPU implementation was found to range from 18x to 33x with the largest number of examples.

Results - Component (Cont.)

- **Lucas-Kanade Tracking Evaluations**



With the largest number of data points the GPU implementation had a speedup of 7x.

PENNSTATE ARL

Works Cited

- Z. Kalal, "Tracking Learning Detection," Center for Vision, Speech and Signal Processing, Faculty of Engineering and Physical Sciences, University of Surrey, 2011.
- P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Conference on Computer Vision and Pattern Recognition, 2001.
- B. Bilgic, B. K. Horn and I. Masaki, "Efficient Integral Image Computation on the GPU," in IEEE Intelligent Vehicle Symposium, 2010.
- J.-Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker- Description of the Algorithm," Microprocessor Research Labs, Intel Corporation, 2000.

Works Cited (Cont.)

- E. Maggio, E. Piccardo, C. Regazzoni and A. Cavallaro, "Particle phd filtering for multi-target visual tracking," in IEEE International Conference on Acoustics, Speed and Signal Processing, Honolulu, 2007.
- D. Gale and L. S. Shapley, "College Admissions and the Stability of Marriage," The American Mathematical Monthly, vol. 69, pp. 9-15, 1962.
- C. Cortes and V. Vapnik, "Support-Vector Networks," AT&T Labs- Research, 1995.
- G. Nebehay, "Robust Object Tracking Based on Tracking-Learning-Detection," The Vienna University of Technology, 2012.

Works Cited (Cont.)

- J. Hoberock and N. Bell, "Thrust Parallel Algorithms Library," [Online]. Available: http://thrust.github.io. [Accessed February 2014].
- A. Ozerov, J.-R. Vigouroux, L. Chevallier and P. Pérez, "On evaluating face tracks in movies," in IEEE International Conference on Image Processing, 2013.
- W. Allen, Director, Hannah and Her Sisters. [Film]. United States: Robert Greenhut, 1986