# Clustering and Synchronizing Multi-Camera Video via Audio Fingerprinting

Nicholas J. Bryan, Paris Smaragdis*, and Gautham Mysore*

Stanford University | CCRMA
*Advanced Technology Labs | Adobe Systems

CCRMA DSP Seminar, November 13th 2012

# Outline

I Introduction

II Proposed Method

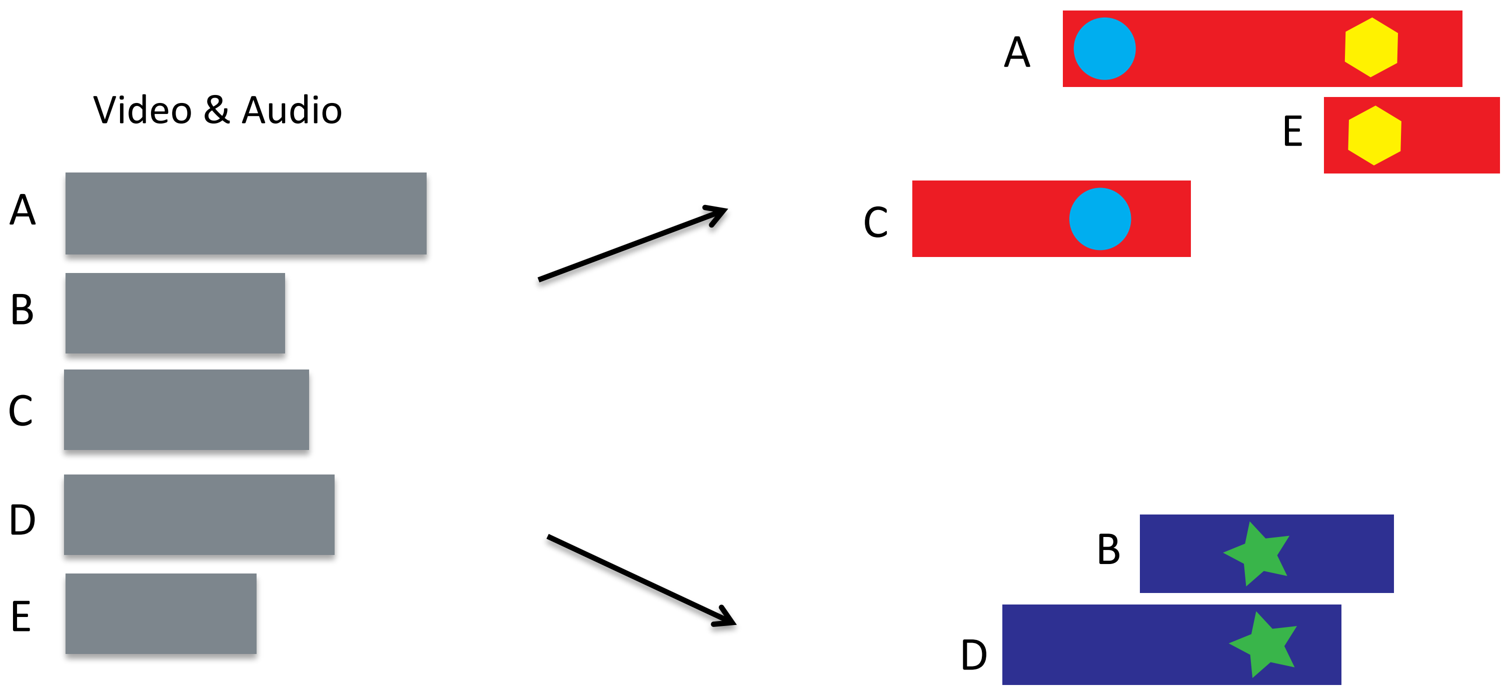
- Non-Linear Transform
- Time-Difference-Of-Arrival Estimation
- Clustering
- Synchronization Refinement
- Efficient Computation

III Evaluation

IV Conclusions

# Introduction

- Identify and synchronize multiple videos of the same event

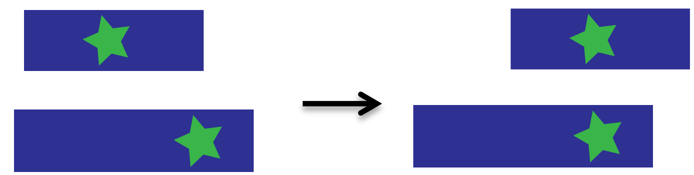


# Motivation

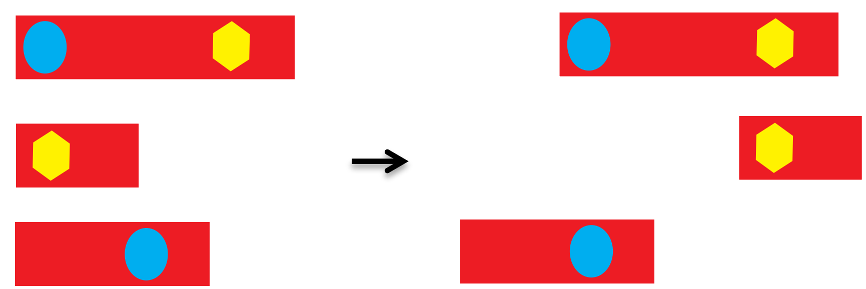
- Proliferation of mobile devices
- Multiple videos of a single event common
  - Moments in history
  - Weddings, concerts, speeches, film sets
- Desired to easily edit video together
  - Grouping/Clustering (Manual)
  - Synchronization (Manual, Hardware)

# Traditional Video Capture

- Dual System Workflow
  - 1 Videographer
  - 1 Sound Engineer



- Multi-Camera Workflow
  - 2+ Videographer
  - 1+ Sound Engineer



# Crowd-Sourced Multi-Camera Video

- 1 Wedding
  - ≈ 300 guest
  - ≈ 100 smartphones/cameras
  - ≈ 10+ videos of "I do"
- 1 concert
  - ≈ 15,000 people
  - ≈ 5,000 smartphones
  - ≈ 100+ of video clips/song
  - ≈ 1000+ video clips/concert
- 1 presidential speech
  - ≈ 200,000 people
  - ≈ 70,000 smartphones
  - ≈ 10,000+ videos

# Demo Video

- Taylor Swift’s “Fearless”

# Outline

I Introduction

II Proposed Method

- Non-Linear Transform
- Time-Difference-Of-Arrival Estimation
- Clustering
- Synchronization Refinement
- Efficient Computation

III Evaluation

IV Conclusions

# General Approach

- Use audio
  - Typically more "global"
  - Allows visually disjoint video
- Time-difference-of-arrival estimation
  - For each pair of clips in collection, compute time offset which best synchronizes the given pair using standard correlation
  - Use correlation signals to decide if the two files should match or not

# Problems

- Computationally expensive
- No accurate (straightforward) clustering method
- Not robust

# Audio Fingerprinting

- Short-duration signatures via feature extraction
- Finds identical (or similar) matches of unknown clip with DB
- Hash fingerprints for fast search and retrieval
- Shazam, SoundHound, Philips, Gracenote, etc.
- See [Wang 2003] & [Haitsma and Kalker 2003]

# Audio Fingerprinting for Multi-Camera

- Slightly different problem
  - Group all clips in DB (multiple matching)
  - Time synchronize all clips within each group

- Audio-fingerprinting for multi-camera
  - Principal of most methods yield sync offset
  - Robust and fast!
  - Initial work over the last few years

    [Shrestha et al. 2007] & [Kennedy and Naaman 2009]

# Proposed Method

1. Non-Linear Transform (Fingerprinting Step)
2. Time-Difference-Of-Arrival Estimation
3. Clustering
4. Synchronization Refinement
5. Efficient Computation

# Outline

I Introduction

II Proposed Method

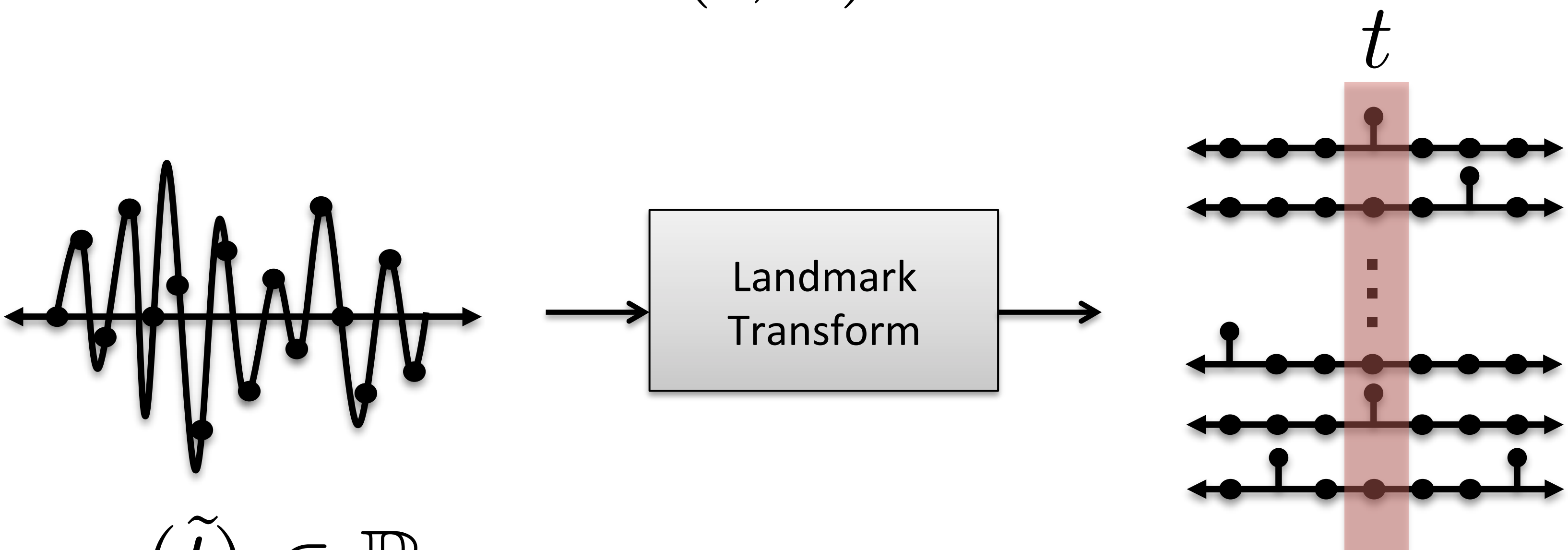
- Non-Linear Transform
- Time-Difference-Of-Arrival Estimation
- Clustering
- Synchronization Refinement
- Efficient Computation

III Evaluation

IV Conclusions

# Non-Linear (Landmark) Transform

- Convert time-domain audio signal $x(\tilde{t})$ into a high-dimensional, sparse, binary landmark signal $\mathbf{L}(t, h)$



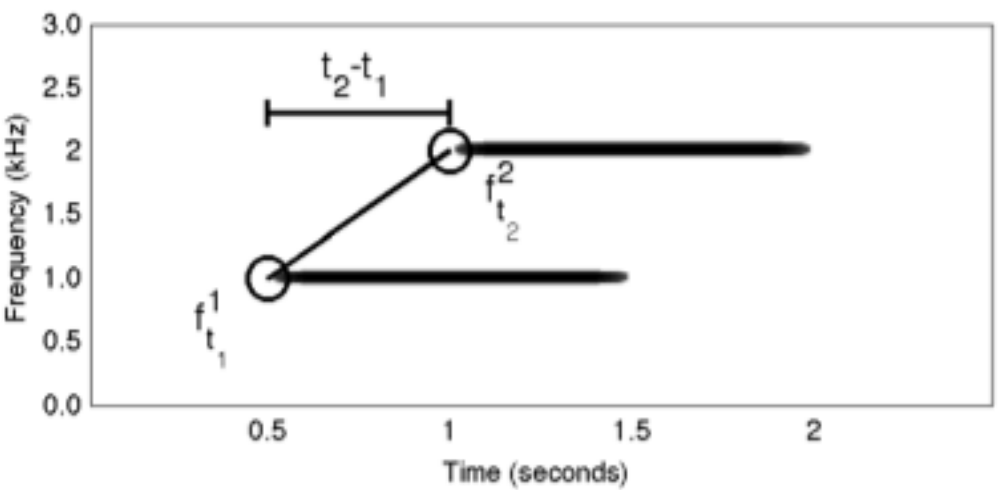

$x(\tilde{t}) \in \mathbb{R}$

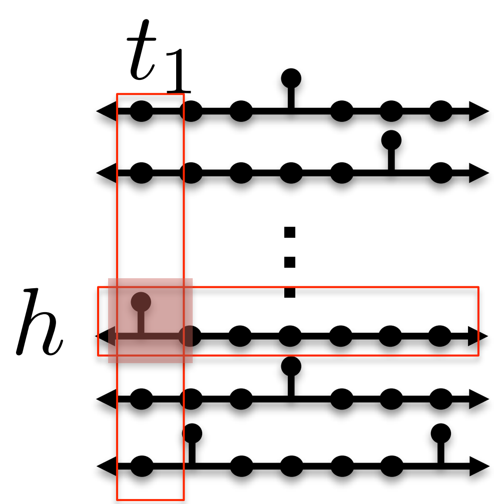
$\mathbf{L}(t, \cdot) \in \{0, 1\}^N$

# Landmarks

- Spectral peak pairs as landmarks [Wang 2003]
  – Short-time Fourier transform
  – Landmark = [f1, f2, Δt] + absolute time offset
  – Place each landmark in appropriate location in $\mathbf{L}(t,h)$

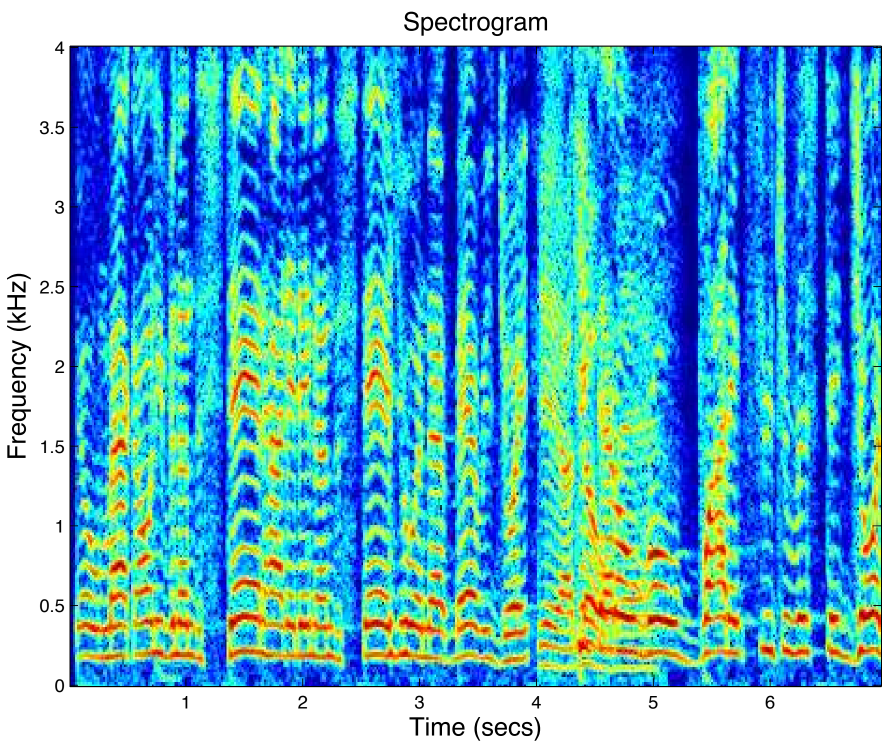
$$(t_1, h = [f^1_{t_1}, f^2_{t_2}, t_2 - t_1]) \qquad \mathbf{L}(t_1, h) = 1$$

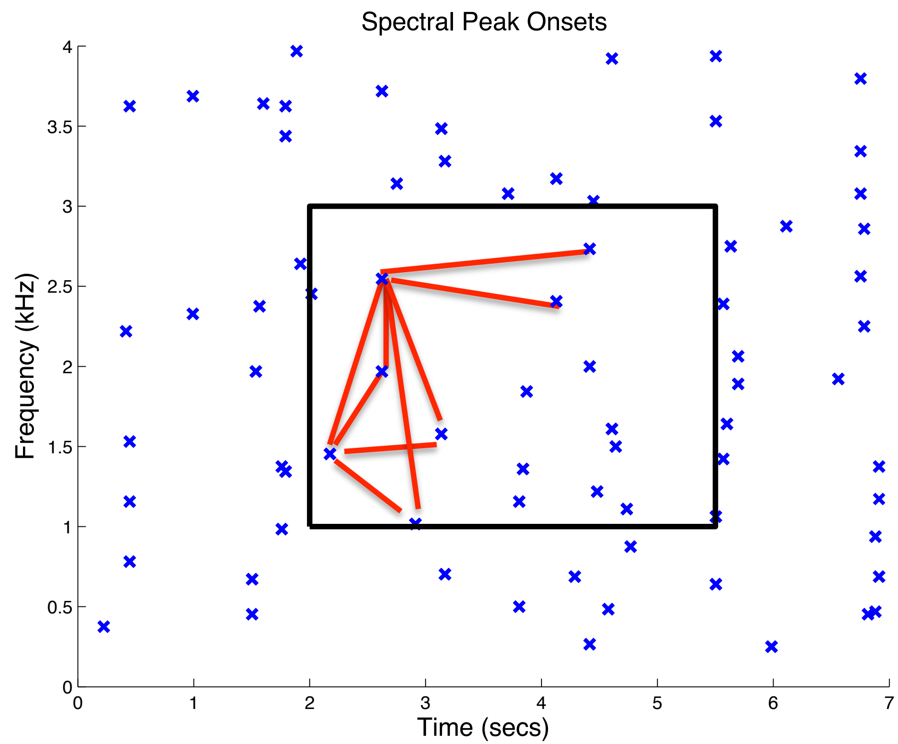


# Landmarks as Constellations

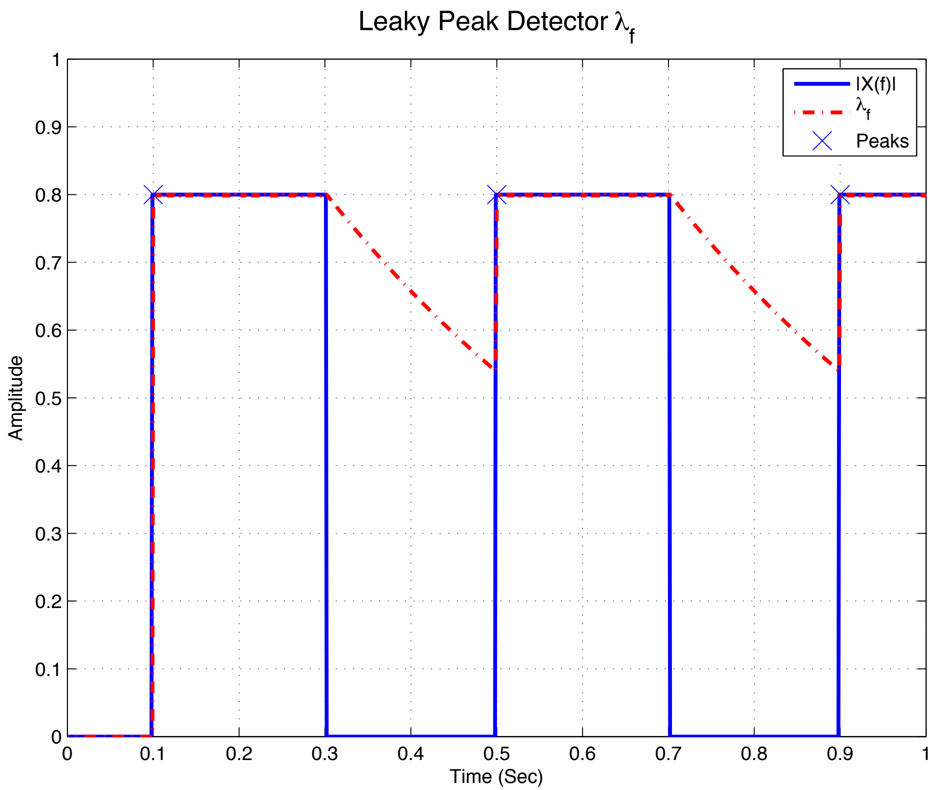
- With a large number of peaks, peak pairs are created in a limited time-frequency range



# Simple Frequency Peak Detector

- Short-time Fourier transform
- Leaky integrator peak detector for each FFT bin




Leaky Peak Detector

$$\text{if } \; |X(f)| > \hat{\lambda}_f$$

$$\hat{\lambda}_f = |X(f)| \quad \text{//Peak Onset}$$

$$\text{else}$$

$$\hat{\lambda}_f = \hat{\lambda}_f - (1 - e^{-1/(\tau_f f_s)})\hat{\lambda}_f$$

# Outline

I Introduction

II Proposed Method

- Non-Linear Transform
- Time-Difference-Of-Arrival Estimation
- Clustering
- Synchronization Refinement
- Efficient Computation

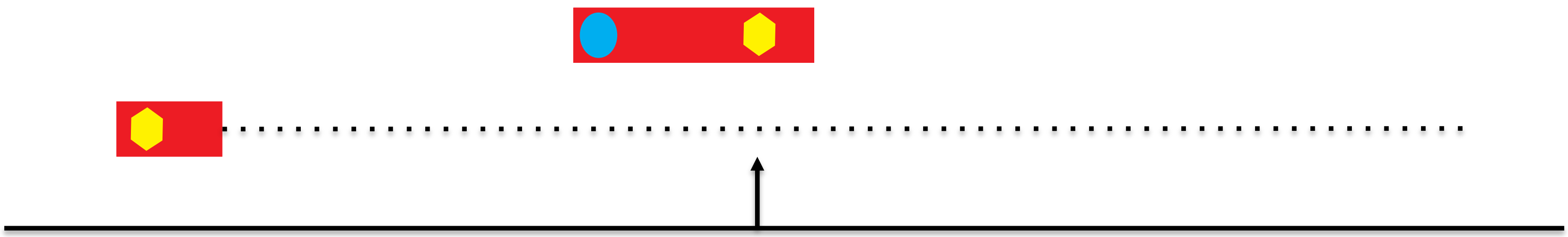
III Evaluation

IV Conclusions

# Time-Difference-Of-Arrival Estimation

- Pairwise cross-correlation method
  - Correlate each track with each other
  - Find argmax for offset
  - i.e. Matched filter

$$R_{ij}(t) = \sum_{\tau=-\infty}^{\infty} x_i(\tau) x_j(t+\tau)$$



# Landmark Cross-Correlation

- Landmark cross-correlation

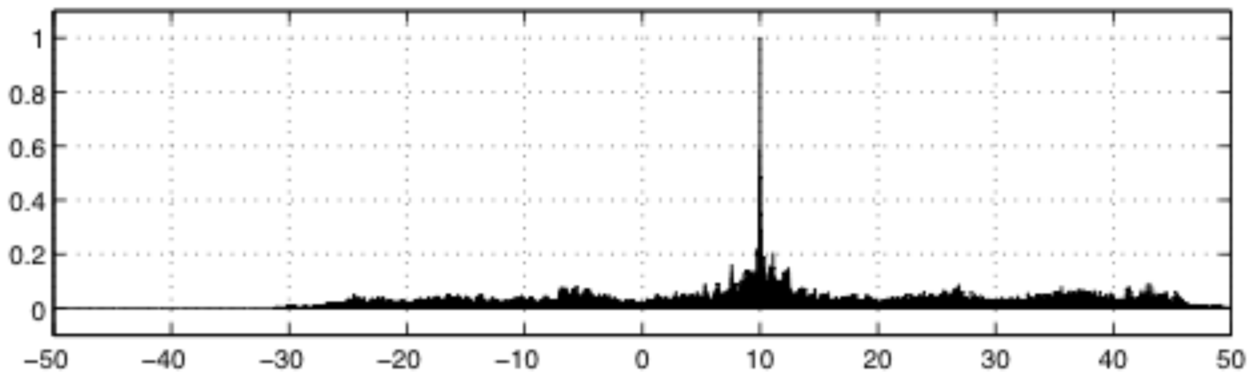
$$R_{\mathbf{L}_i,\mathbf{L}_j}(t) = \sum_{\tau=-\infty}^{\infty} \mathbf{L}_i(\tau)^{\mathrm{T}} \mathbf{L}_j(t+\tau)$$

- Time-Difference-Of-Arrival Estimation

$$\hat{t}_{ij} = \arg\max_t \; R_{\mathbf{L}_i,\mathbf{L}_j}(t)$$

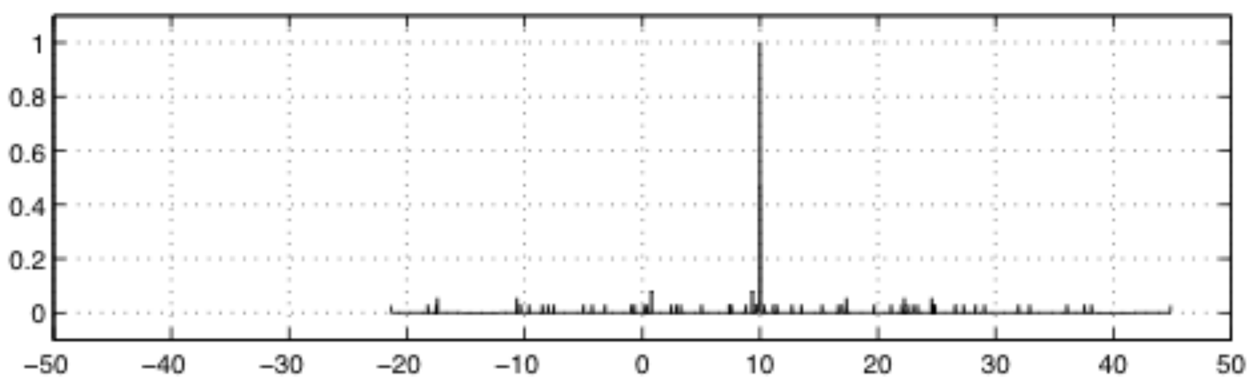
# Time-Difference-Of-Arrival Estimation

$$R_{x_i,x_j}(t)$$




(a) Normalized absolute time-domain cross-correlation.

$$R_{\mathbf{L}_i,\mathbf{L}_j}(t)$$




(b) Normalized landmark cross-correlation.

# Outline

I Introduction

II Proposed Method

- Non-Linear Transform

- Time-Difference-Of-Arrival Estimation

- Clustering

- Synchronization Refinement

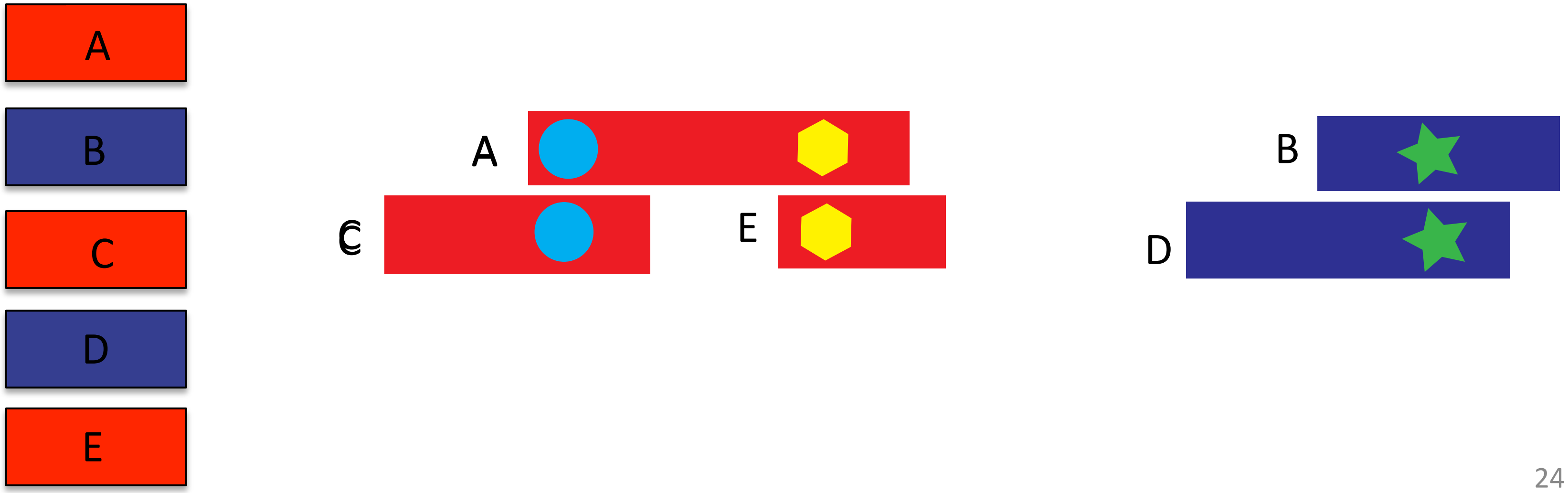
- Efficient Computation

III Evaluation

IV Conclusions

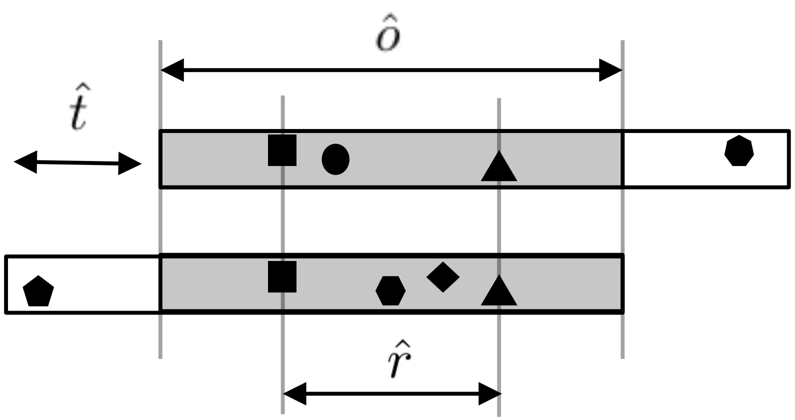
# Clustering

- Agglomerative Clustering
  - Initialize each clip as a separate cluster and merged into successively larger clusters
  - Merge most confidence matches first
- Confidence as function of stats from best potential sync
- Reject unconfident merges based on decision rules



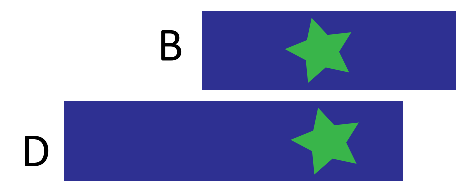
# Merge Decision Rules

- Maximum of correlation
- Mean and variance of cross-correlation
- Percentage of total matching landmarks in the overlap region $\hat{o}$
- Overall time range $\hat{r}$ defined by the set of matching landmarks
- Overlap region $\hat{o}$ length
- Ignore overly common landmarks (i.e. 60Hz)



# Clustering Output

- Groups w/pairwise sync offset and confidence scores

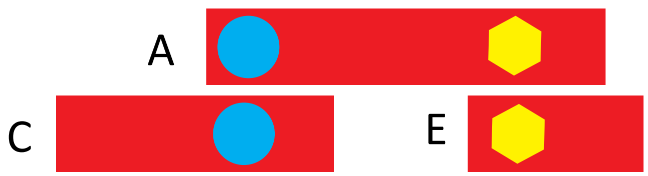



Offsets (seconds)

| $\hat{t}$ | $B$ | $D$ |
|---|---|---|
| $B$ | 0 | -11.5 |
| $D$ | 11.5 | 0 |

Confidence Score

| $\hat{S}$ | $B$ | $D$ |
|---|---|---|
| $B$ | - | 23 |
| $D$ | 23 | - |




Offsets (seconds)

| $\hat{t}$ | $A$ | $C$ | $E$ |
|---|---|---|---|
| $A$ | 0 | -5 | 10 |
| $C$ | 5 | 0 | - |
| $E$ | -10 | - | 0 |

Confidence Score

| $\hat{S}$ | $A$ | $C$ | $E$ |
|---|---|---|---|
| $A$ | - | 30 | 20 |
| $C$ | 30 | - | - |
| $E$ | 20 | - | - |

# Outline

I Introduction

II Proposed Method

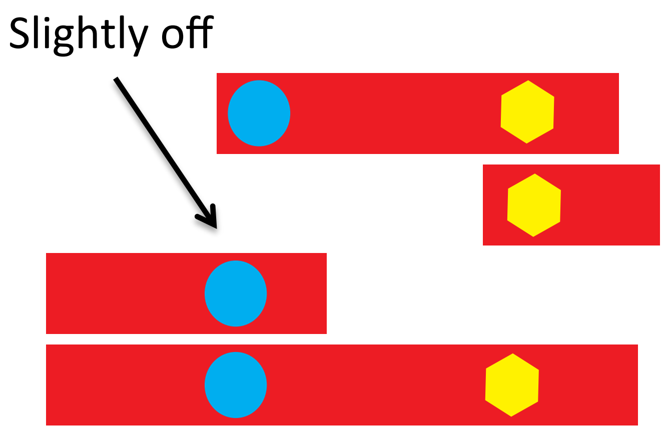
- Non-Linear Transform
- Time-Difference-Of-Arrival Estimation
- Clustering
- Synchronization Refinement
- Efficient Computation

III Evaluation

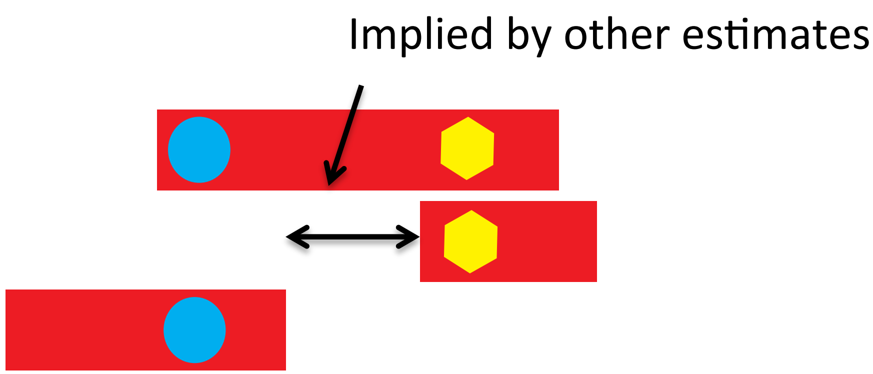
IV Conclusions

# Synchronization Refinement

- Refinement is required for clusters of three or more if:
  1. Inconsistent pairwise TDOA estimates do not satisfy all triangle equalities $\hat{t}_{AC} \neq \hat{t}_{AB} + \hat{t}_{BC}$ within a cluster
  2. One or more TDOA estimates within any cluster is unknown caused by non-overlapping clips




(a) Case 1




(a) Case 2

# Greedy Match-and-Merge

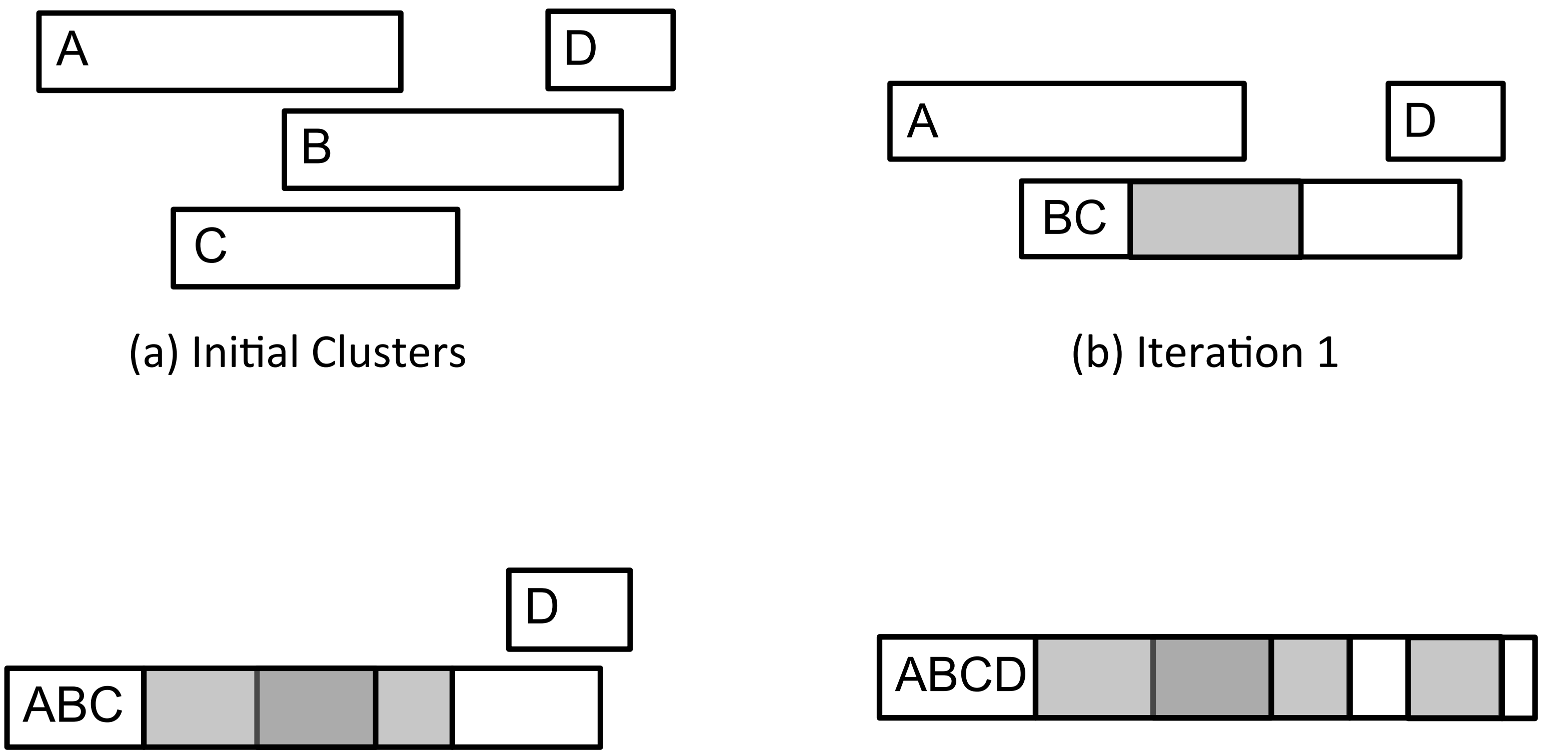
1. Find the most confident TDOA estimate $\hat{t}_{ij}$ within the cluster in terms of $\hat{R}_{\mathbf{L}_i,\mathbf{L}_j}$ or similar confidence score.

2. Merge the landmark signals $\mathbf{L}_i$ and $\mathbf{L}_j$. First time shift $\mathbf{L}_j$ by $\hat{t}_{ij}$ and then multiply or add the two signals together (depending on the desired effect).

3. Update the remaining TDOA estimates and confidence scores to respect the file merge.

4. Repeat until all files within the cluster are merged.

# Greedy Match-and-Merge Graphically




(a) Initial Clusters (b) Iteration 1

(c) Iteration 2 (d) Iteration 3

# Outline

I Introduction

II Proposed Method

- Non-Linear Transform

- Time-Difference-Of-Arrival Estimation

- Clustering

- Synchronization Refinement

- Efficient Computation

III Evaluation

IV Conclusions

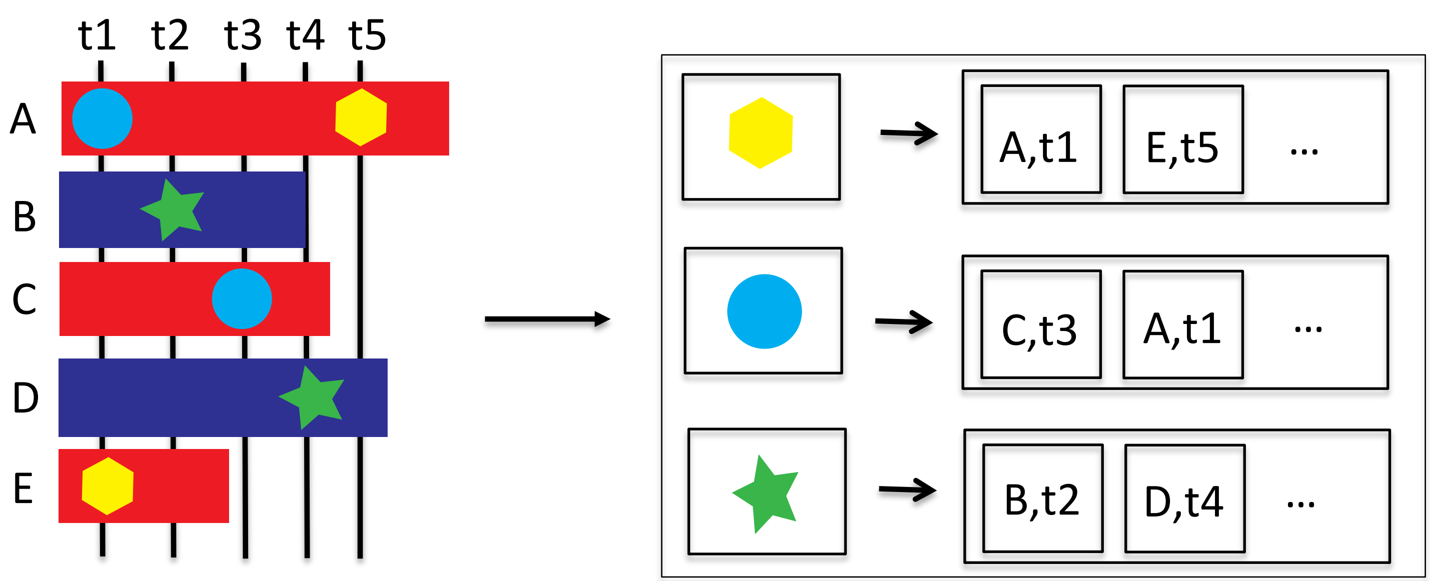
# Efficient Computation

- Leverage knowledge of landmark signal and perform “sparse” cross-correlation in a special way (fingerprinting)

- Use some form of associative array, map, or dictionary to store landmarks and compute all pairwise correlations
  - Direct arrays
  - Binary tree
  - Hash table

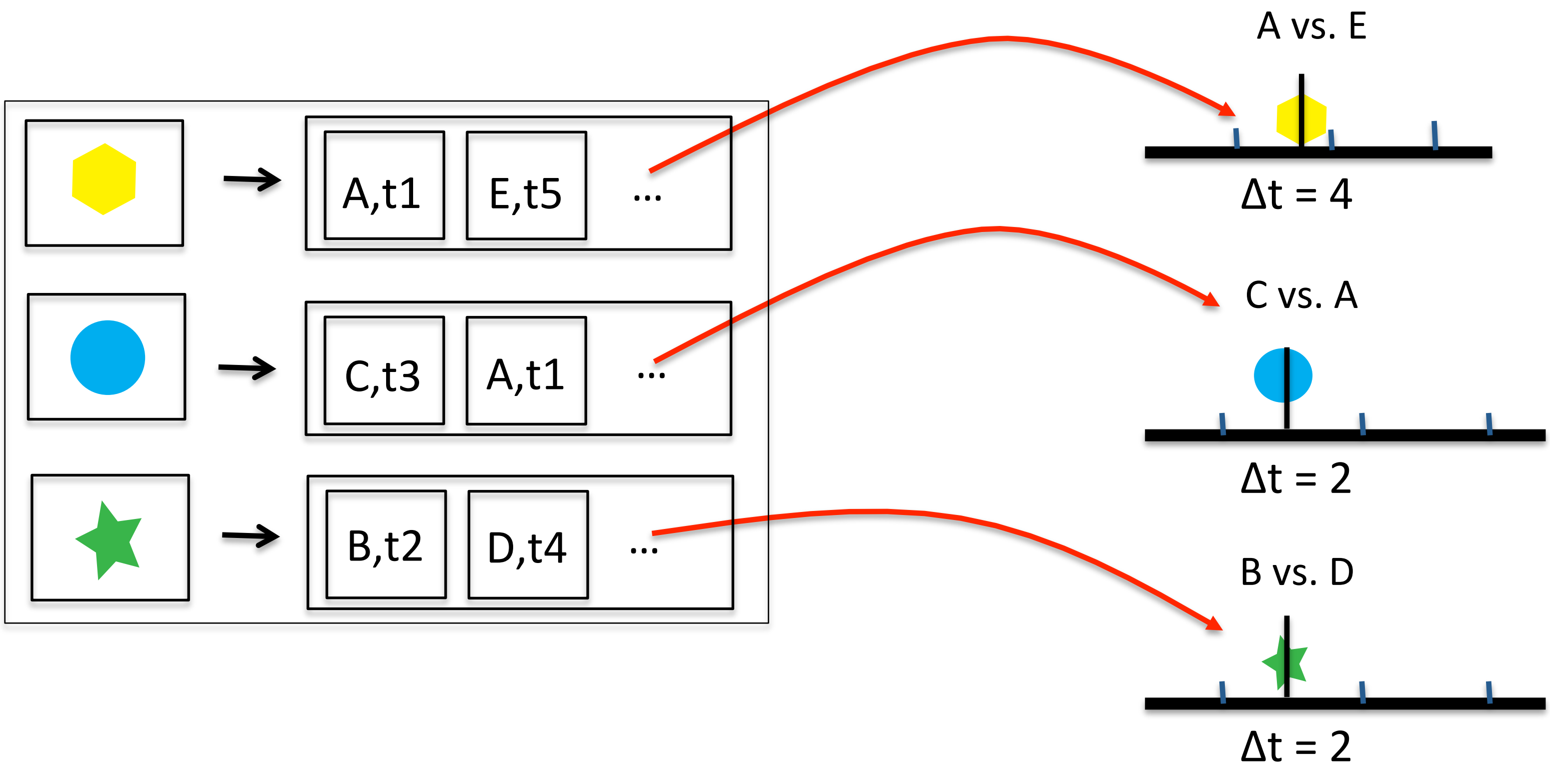
# Map Structure I

- Create map structure of all landmarks
  - Key = (*f1*, *f2*, *Δt*)
  - Value = (*FileID*, *AbsoluteTimeOffset*)
- Matching files will have identical landmark
- Difference between *AbsoluteTimeOffset* of gives sync



# Map Structure II

- Convert map structure to pairwise correlations
- For each landmark, compute all pairwise time differences and store in the appropriate pairwise correlation



# General Computational Benefit

- Naïve pairwise correlations
  1. $\frac{P!}{2(P-2)!}$ pairwise correlations, $P =$ number of files
  2. Each correlation $O(N\log(N))$, $N =$ samples in file

- Drastically reduces the computational cost
  1. Eliminates pairwise correlations for clips that don't match
  2. Makes each pairwise correlation faster

- Computes correlation computation for only the salient parts (landmarks) of audio

# Ideal Case

1. Pairwise Comparisons
   - All landmarks are unique its group
   - Only performs pairwise correlations within each group
   - For large # groups/small # clips, this is savings huge

2. Single pairwise correlation
   - Only correlate points with matching landmarks, no computation for 0s
   - Ideal case with no false positive matches results in a $O(M)$ cost, with $M$ = number of matching landmarks

# Outline

I Introduction

II Proposed Method

- Non-Linear Transform

- Time-Difference-Of-Arrival Estimation

- Clustering

- Synchronization Refinement

- Efficient Computation

III Evaluation

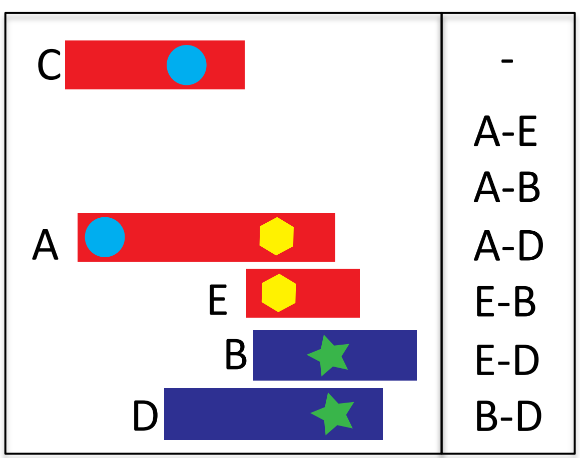
IV Conclusions

# Evaluation Metrics

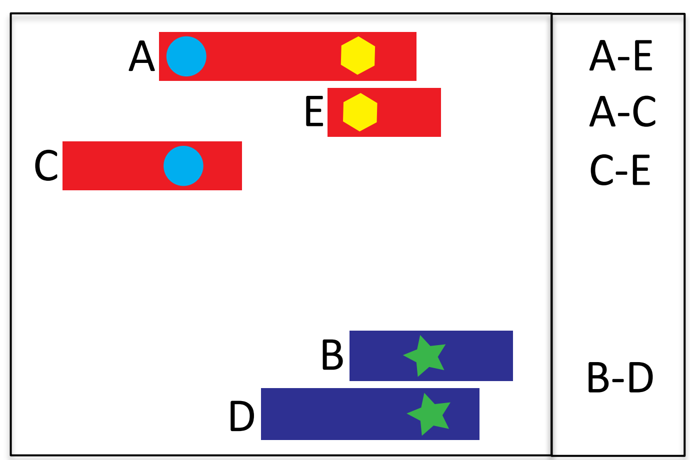
- Performance measures
  - Precision, Recall, F1 score
  - Computed on pairwise matches of final clusters
- Computational cost
  - Compute time (seconds)
  - Throughput (seconds processed/seconds of compute time)
- Benchmark
  - Comparison to commercial multi-camera software Plural Eyes

# Precision, Recall, and F1

- Precision
  - fraction of estimated pairwise merges retrieved that are correct
- Recall
  - fraction of correct pairwise merges retrieved
- F1 score
  - harmonic mean of precision and recall $2PR/(P+R)$



Estimated Clusters



Ground Truth Clusters

P = 2/6
R = 2/5
F1 = 8/22

# Datasets

- Speech (180 clips from film set)
  - Average length 20-40 seconds
  - 54 clusters of one file
  - 54 clusters of two files
  - 6 clusters of three files

- Music (23 clips from live music concerts)
  - Average length 3-5 minutes
  - 1 cluster of 7 files
  - 2 clusters of 8 files

All audio files are downsampled to common sample rate of 8kHz for efficiency

# Precision, Recall, and F1 Results

| | Speech | Music | Speech + Music |
|---|---|---|---|
| **Precision** | 100.0 % | 100.0 % | 100.0 % |
| **Recall** | 97.0 % | 100.0 % | 99.2 % |
| **F-Score** | 98.5 % | 100.0 % | 99.6 % |

(a) Precision, recall, and $F_1$-scores.

- As expected from using the feature extraction of [Wang 2003]

# Computational Cost

| | **Speech** | **Music** | **Speech + Music** |
|---|---|---|---|
| **Proposed** | 47.0 | 41.1 | 90.1 ≈ linear |
| **Traditional** | 1550 | 197 | 3600 not linear |

(a) Computation time (s).

| | **Speech** | **Music** | **Speech + Music** |
|---|---|---|---|
| **Proposed** | 164.6 | 146.5 | 152.7 |
| **Traditional** | 5.0 | 30.5 | 3.9 |

(b) Throughput (s/s).

Rough Timing on MacBook Pro, OSX 10.6.8, 2.66 GHz Intel Core i7, unoptimized C++.

# Benchmark (Speech Dataset)

- Accuracy Measures
  - Proposed method F1 ≈ 99%
  - Plural Eyes 2.1.0 F1 ≈ 95%

- Computational Cost
  - Proposed method ≈ 3 minutes
  - Plural Eyes 1.2.0 ≈ 6 hours
  - Plural Eyes 2.1.0 ≈ 2 hours
  - Plural Eyes 2.1.0 (hard) ≈ 10 hours

# Outline

I Introduction

II Proposed Method

- Non-Linear Transform
- Time-Difference-Of-Arrival Estimation
- Clustering
- Synchronization Refinement
- Efficient Computation

III Evaluation

IV Conclusions

# Future Work & Research Directions

- Video analog to photo “stitching”
  - Crowd-sourced multi-camera video
  - Easily change both video and audio viewpoint

- Denoising/improving audio quality from groups

- Spatial audio processing
  - Use for time delay estimation
  - Large-scale beamforming, directional listening, etc.

# Conclusions

- Method of clustering and sync of multi-camera videos using audio
  - Non-Linear Transform
  - Time-Difference-Of-Arrival Estimation
  - Clustering
  - Synchronization Refinement
  - Efficient Computation
- Fast and accuracy

# References


- Jaap Haitsma and Ton Kalker, "A Highly Robust Audio Fingerprinting System With an Efficient Search Strategy," Journal of New Music Research , vol. 32, no. 2, 2003.
- A.L. Wang, "An Industrial-Strength Audio Search Algorithm," in Proc. 4th Int. Symposium on Music Information Retrieval (ISMIR) , October 2003.
- P. Shrestha, M. Barbieri, and H. Weda, "Synchronization of multi-camera video recordings based on audio," in Proc. 15th Intl. Conf. on Multimedia , 2007.
- L. Kennedy and M. Naaman, "Less talk, more rock: automated organization of community-contributed collections of concert videos," in Proc. 18th Int. Conf. on World Wide Web , 2009.
- D. Ellis (2009). "Robust Landmark-Based Audio Fingerprinting", http://labrosa.ee.columbia.edu/matlab/fingerprint

# Demo Video

- Dave Matthews Band's "Everyday"

# Clustering and Synchronizing Multi-Camera Video via Audio Fingerprinting

Nicholas J. Bryan, Paris Smaragdis*, and Gautham Mysore*

Stanford University | CCRMA
*Advanced Technology Labs | Adobe Systems

CCRMA DSP Seminar, November 13th 2012