Kobe University and Muroran Institute of Technology at TRECVID 2012 Semantic Indexing Task

– Fast and Exact Processing of Large-scale Video Data based on Matrix Operation –

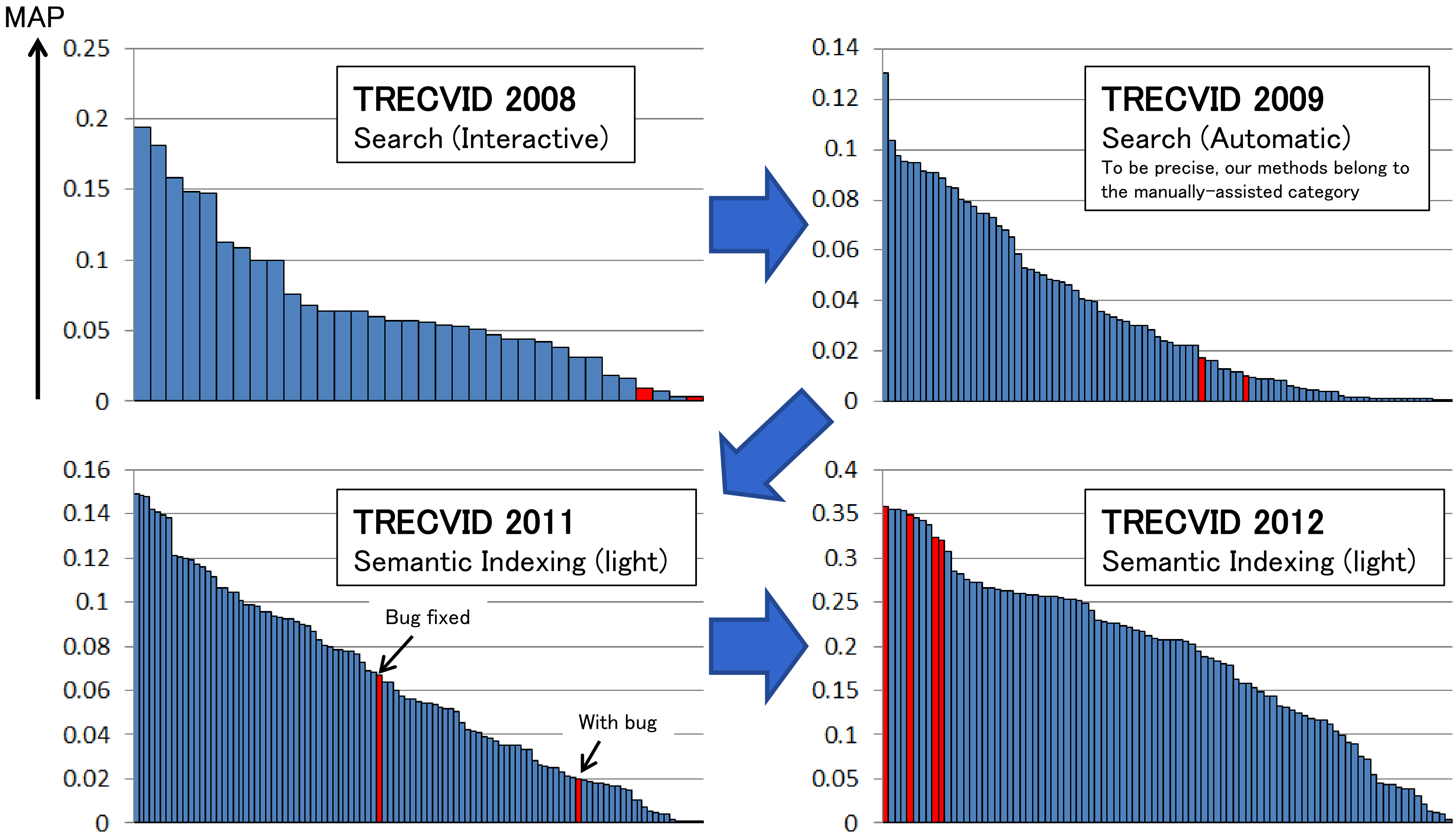
Kimiaki Shirahama

Muroran Institute of Technology

Kuniaki Uehara

Kobe University

Our TRECVID History



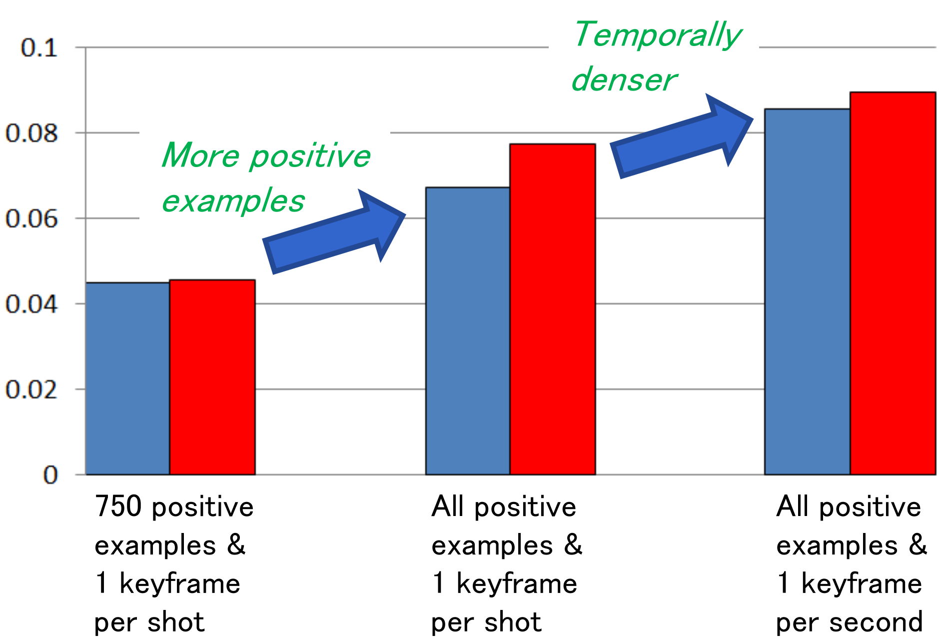
We achieved the highest MAP in TRECVID 2012 SIN (light) task!

Lessons That We Learned

To build accurate concept detectors,

- A large number of training examples are needed
- Features densely sampled in both the spatial and temporal dimensions are needed
  (*spatially-temporally dense features*)

MAPs for 23 concepts in TRECVID 2011 SIN (light)



SIFT descriptors on Harris-Laplace detector

Spatially denser

SIFT descriptors on dense sampling

High computational cost is required to process many training examples and spatially-temporally dense features!

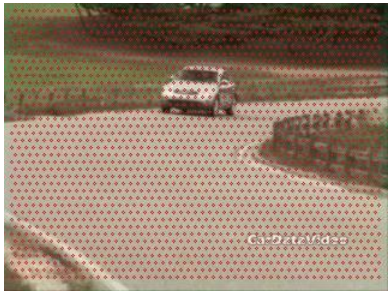
Our Goal in TRECVID 2012

Fast processing of large-scale video data
- Approximation (or simpler) methods degrade the detection performance
- Parallelization using multiple processors or GPUs requires expensive hardware resources

→ Develop a fast and exact method on a single processor
Not process data one by one, but process them in batch based on matrix operation

1. Fast SVM training/test based on ***batch computation of kernel values***
2. Fast spatially-temporally dense feature extraction based on ***batch computation of probability densities***

→ Shot representation considering millions of feature descriptors



3. Diversity of a concept's appearances

Bagging: Fuse many detectors built using different sets of training examples
← Owing to our fast SVM training/test method

Motivating Example (1/3)
– Euclidian Distance Computation –

Compute the Euclidian distance between each pair of N examples $\boldsymbol{x}_i$ ($\boldsymbol{D}$–dimensional)

$$\begin{bmatrix} dist(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & dist(\boldsymbol{x}_1, \boldsymbol{x}_j) & \cdots & dist(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & & \vdots & & \vdots \\ dist(\boldsymbol{x}_i, \boldsymbol{x}_1) & \cdots & dist(\boldsymbol{x}_i, \boldsymbol{x}_j) & \cdots & dist(\boldsymbol{x}_i, \boldsymbol{x}_N) \\ \vdots & & \vdots & & \vdots \\ dist(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & dist(\boldsymbol{x}_N, \boldsymbol{x}_j) & \cdots & dist(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix}$$

$$dist(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{d=1}^{D} (x_{id} - x_{jd})^2$$

Naive implementation

Set the i-th and j-th examples ←

Compute the squared difference in each dimension ←

```
for i = 1 → N do
    for j = 1 → N do
        dist[i][j] = 0
        for d = 1 → D do
            dist[i][j] += (x_id − x_jd)^2
        end
    end
end
```

Too slow!

Motivating Example (2/3)
– Euclidian Distance Computation –

$$dist(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{d=1}^{D}(x_{id} - x_{jd})^2 = \sum_{d=1}^{D}(x_{id})^2 + \sum_{d=1}^{D}(x_{jd})^2 - 2\sum_{d=1}^{D} x_{id}x_{jd}$$

Matrix operation

$\boldsymbol{x}_i$

$$\begin{bmatrix} x_{11} & \cdots & x_{i1} & \cdots & x_{N1} \\ \vdots & & \vdots & & \vdots \\ x_{1D} & \cdots & x_{iD} & \cdots & x_{ND} \end{bmatrix}$$

Take the square of each element

$$\begin{bmatrix} x_{11}^2 & \cdots & x_{i1}^2 & \cdots & x_{N1}^2 \\ \vdots & & \vdots & & \vdots \\ x_{1D}^2 & \cdots & x_{iD}^2 & \cdots & x_{ND}^2 \end{bmatrix}$$

Compute the sum of elements in each column

$$\begin{bmatrix} \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{id})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 \end{bmatrix}$$

Motivating Example (3/3)
– Euclidian Distance Computation –

$$dist(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{d=1}^{D}(x_{id} - x_{jd})^2 = \sum_{d=1}^{D}(x_{id})^2 + \sum_{d=1}^{D}(x_{jd})^2 - 2\sum_{d=1}^{D} x_{id}x_{jd}$$

Matrix operation

$$\begin{bmatrix} \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{id})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 \\ \vdots & & \vdots & & \vdots \\ \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{id})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 \end{bmatrix} + \begin{bmatrix} \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{id})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 \end{bmatrix}$$

Create N copies along the row direction

Motivating Example (3/3)
– Euclidian Distance Computation –

$$dist(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{d=1}^{D}(x_{id} - x_{jd})^2 = \sum_{d=1}^{D}(x_{id})^2 + \sum_{d=1}^{D}(x_{jd})^2 - 2\sum_{d=1}^{D} x_{id}x_{jd}$$

Matrix operation

$$\begin{bmatrix} \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{id})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 \\ \vdots & & \vdots & & \vdots \\ \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{id})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 \end{bmatrix} + \begin{bmatrix} \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{1d})^2 \\ \vdots & & \vdots & & \vdots \\ \sum_{d=1}^{D}(x_{Nd})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 \end{bmatrix}$$

Create N copies along the row direction

Create N transposed copies along the column direction

$$- 2 \begin{bmatrix} x_{11} & \cdots & x_{i1} & \cdots & x_{N1} \\ \vdots & & \vdots & & \vdots \\ x_{1D} & \cdots & x_{iD} & \cdots & x_{ND} \end{bmatrix}^T \begin{bmatrix} x_{11} & \cdots & x_{i1} & \cdots & x_{N1} \\ \vdots & & \vdots & & \vdots \\ x_{1D} & \cdots & x_{iD} & \cdots & x_{ND} \end{bmatrix}$$

Motivating Example (3/3)
– Euclidian Distance Computation –

$$dist(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{d=1}^{D} (x_{id} - x_{jd})^2 = \sum_{d=1}^{D} (x_{id})^2 + \sum_{d=1}^{D} (x_{jd})^2 - 2\sum_{d=1}^{D} x_{id} x_{jd}$$

Matrix operation

$$\begin{bmatrix} \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{id})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 \\ \vdots & & \vdots & & \vdots \\ \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{id})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 \end{bmatrix} + \begin{bmatrix} \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{1d})^2 & \cdots & \sum_{d=1}^{D}(x_{1d})^2 \\ \vdots & & \vdots & & \vdots \\ \sum_{d=1}^{D}(x_{Nd})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 & \cdots & \sum_{d=1}^{D}(x_{Nd})^2 \end{bmatrix}$$

Create N copies along the row direction

Create N transposed copies along the column direction

$$- 2 \begin{bmatrix} x_{11} & \cdots & x_{i1} & \cdots & x_{N1} \\ \vdots & & \vdots & & \vdots \\ x_{1D} & \cdots & x_{iD} & \cdots & x_{ND} \end{bmatrix}^T \begin{bmatrix} x_{11} & \cdots & x_{i1} & \cdots & x_{N1} \\ \vdots & & \vdots & & \vdots \\ x_{1D} & \cdots & x_{iD} & \cdots & x_{ND} \end{bmatrix}$$

Computational time comparison
Xeon W5590 3.33GHz, Memory: 24GB
(Each example has 16,384 dimensions)

	1,000 examples	5,000 examples
Naive	200 sec	5,027 sec
Matrix operation	**0.5 sec**	**9.7 sec**

Effectiveness of the batch computation over the one-by-one computation!

Fast SVM Training/Test

Training

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

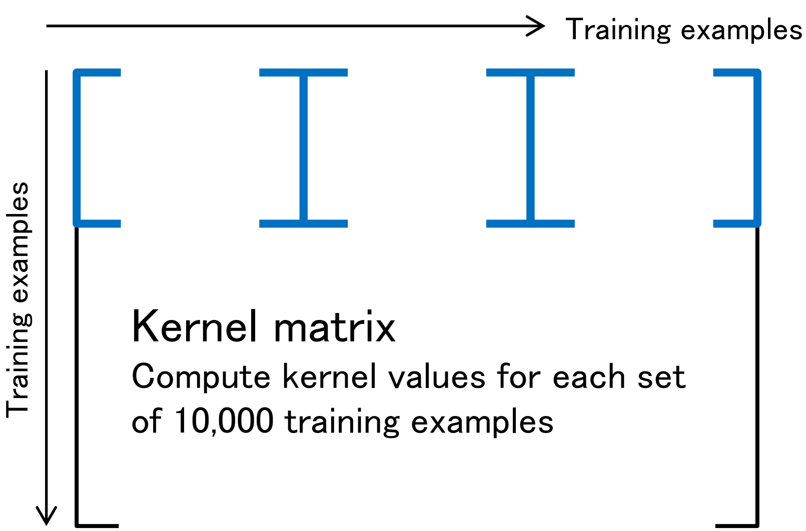
Test

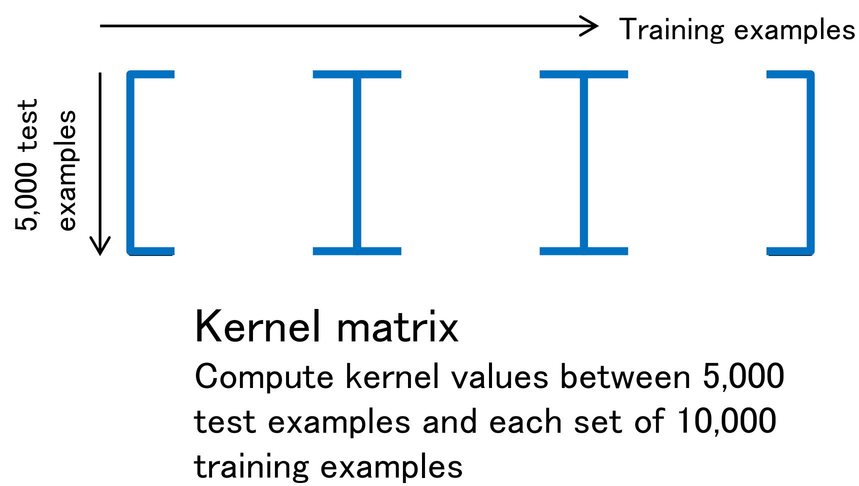
$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) - b$$

RBF kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma dist(\boldsymbol{x}_i, \boldsymbol{x}_j))$

Euclidian distance

Compute in batch kernel values for many training and test examples



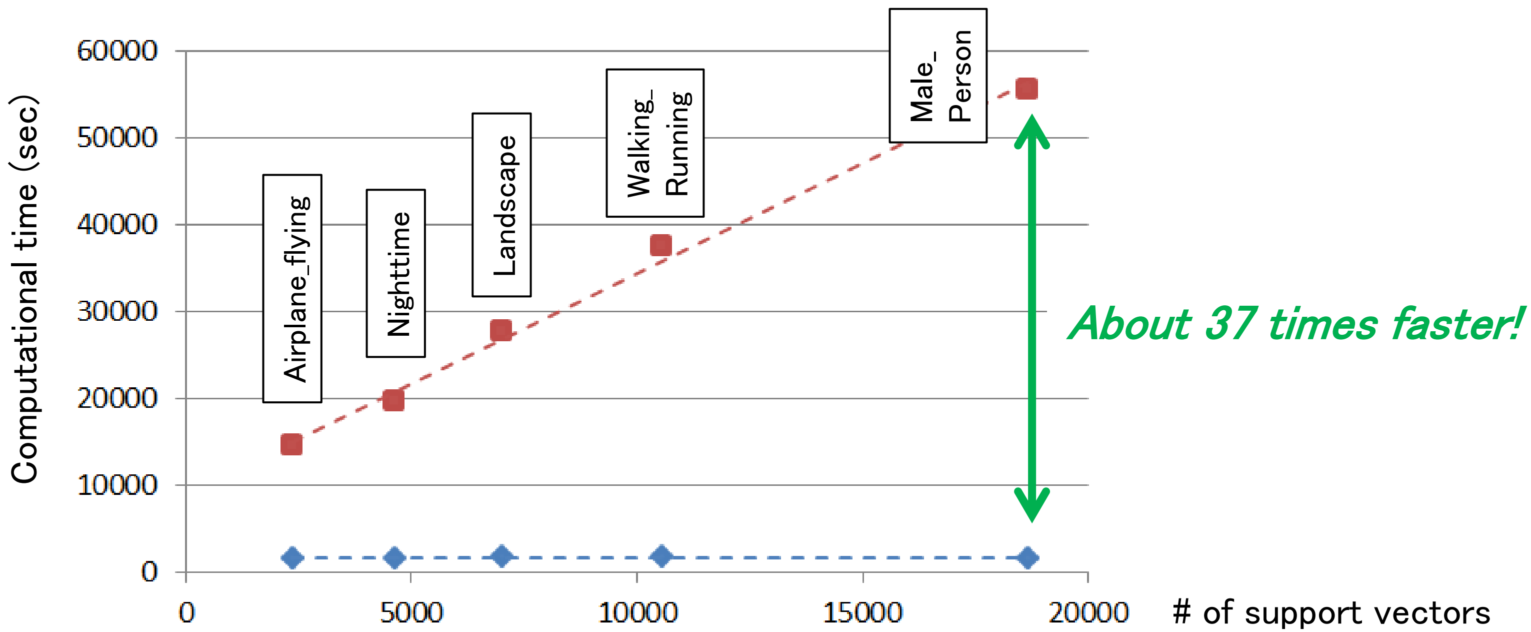


Apply a general SVM solver (LIBSVM precomputed kernel) to kernel matrixes

Efficiency of SVM Training/Test based on Matrix Operation

- Matrix operation: Batch computation of kernel values
- Baseline: One-by-one computation of the kernel value between each pair of examples
  - Training: Kernel values at symmetric positions are computed only once (*i.e.*, $K(x_i,x_j) = K(x_j,x_i)$)
  - Test: kernel values are not computed for training examples, which are not support vectors

→ Computational time linearly increases depending on the number of support vectors



- 30,000 16,834-dimensional training examples (all positive examples, and randomly selected negative examples)
- CPU: Xeon X5690 (3.47GHz)
- MATLAB engine is used to call MATLAB functions in C++ programs
- Data loading time (about 700 sec) is excluded.

GMM-based Supervector Shot Representation (Inoue *et al.*: TMM 2012)

Universal Background Model (UBM):
- Distribution of feature descriptors in the general case
- Extracted using randomly sampled feature descriptors

GMM for a shot:
- Distribution of feature descriptors in the shot

MAP Adaptation: Adopt UBM's means based on maximum a posteriori approach

Adapted mean, UBM's mean, Multivariate normal distribution

$$\hat{\mu_k} = \frac{\tau \bar{\mu_k} + \sum_{i=1}^{N} c_{ik} x_i}{\tau + \sum_{i=1}^{N} c_{ik}} \quad \text{, where} \quad c_{ik} = \frac{\bar{w}_k \mathcal{N}_k(x_i | \bar{\mu_k}, \bar{\Sigma}_k)}{\sum_{k=1}^{K} \bar{w}_k \mathcal{N}_k(x_i | \bar{\mu_k}, \bar{\Sigma}_k)}$$

High computational cost is required to compute probability densities of each feature descriptor x_i for K multivariate normal distributions N_k

Spatially-Temporally Dense RGB SIFT (STD-RGB-SIFT):
RGB SIFT descriptors at every 6th pixel in every other frame
(Sande *et al.*: TPAMI 2010)
→ *The number of descriptors easily reaches millions!*

Fast Spatially-Temporally Dense Feature Extraction

Multivariate normal distribution N_k for a D-dimensional feature descriptor x_i

$$\mathcal{N}_k(x_i|\bar{\mu_k}, \bar{\Sigma_k}) = \frac{1}{\sqrt{(2\pi)^D|\bar{\Sigma_k}|}} e^{-\frac{1}{2}(x_i - \bar{\mu_k})\bar{\Sigma_k}^{-1}(x_i - \bar{\mu_k})}$$

By assuming the independence of dimensions,

$$\mathcal{N}_k(x_i|\bar{\mu_k}, \bar{\Sigma_k}) = \frac{1}{\sqrt{(2\pi)^D \prod_{d=1}^{D} \bar{\sigma}_{kd}}} e^{-\frac{1}{2}\sum_{d=1}^{D} \frac{(x_{id} - \bar{\mu}_{kd})^2}{\bar{\sigma}_{kd}}}$$

Weighted Euclidian distance

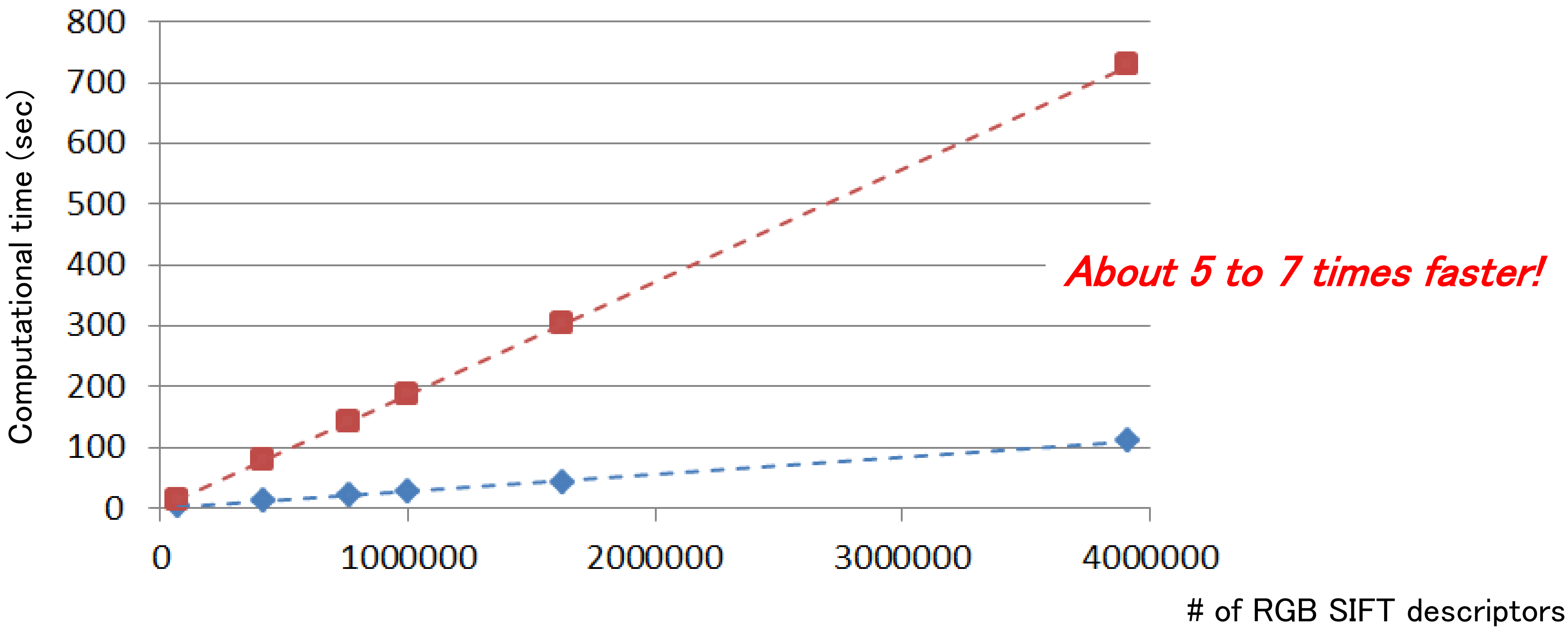
$$\sum_{d=1}^{D} \left(\frac{x_{id}}{\sqrt{\bar{\sigma}_{kd}}} \right)^2 - 2\sum_{d=1}^{D} \frac{x_{id}\bar{\mu}_{kd}}{\bar{\sigma}_{kd}} + \sum_{d=1}^{D} \left(\frac{\bar{\mu}_{kd}}{\sqrt{\bar{\sigma}_{kd}}} \right)^2$$

Extend the batch computation of Euclidian distances to compute in batch probability densities of many feature descriptors for K multivariate normal distributions

(For each set of 100,000 descriptors, we compute their probability densities for 512 distributions in batch)

Efficiency of STD-RGB-SIFT Extraction

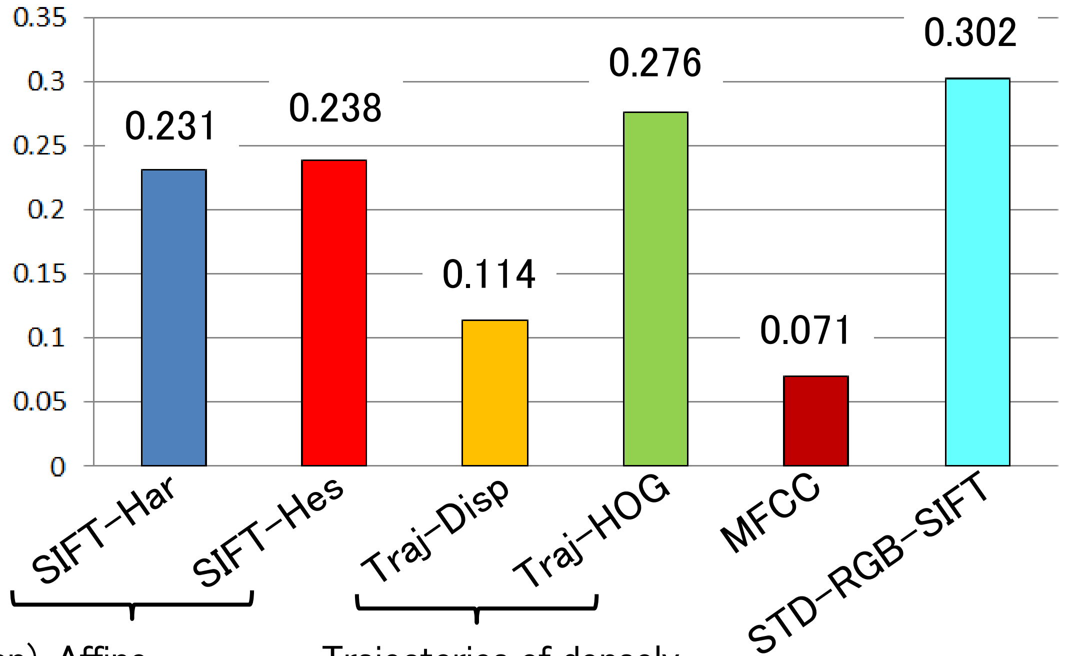
- ◆ Matrix operation: Batch computation of probability densities
- ■ Baseline: One-by-one computation of probability densities based on the weighted Euclidian distance formulation



- CPU: Xeon X5690 (3.47GHz)
- MATLAB engine is used to call MATLAB functions in C++ programs
- Each computational time includes the time required for PCA, where 384-dimensional RGB SIFT descriptors are projected into the space of 32 independent dimensions.

Effectiveness of STD-RGB-SIFT

MAP of SVMs built on each single feature (15 concepts in SIN (light))



Harris(Hessian)-Affine detector for every other frame (Mikolajczyk *et al.*: IJCV 2005)

Trajectories of densely sampled points (Wang *et al.*: CVPR 2011)

STD-RGB-SIFT significantly outperforms the other features!

Fusing Detectors on Different Features

L_A_kobe_muro_l6_1: Weighted linear fusion of 6 SVMs, each built on one feature

– Feature weights are determined by a gradient-ascend approach which maximizes the average precision.

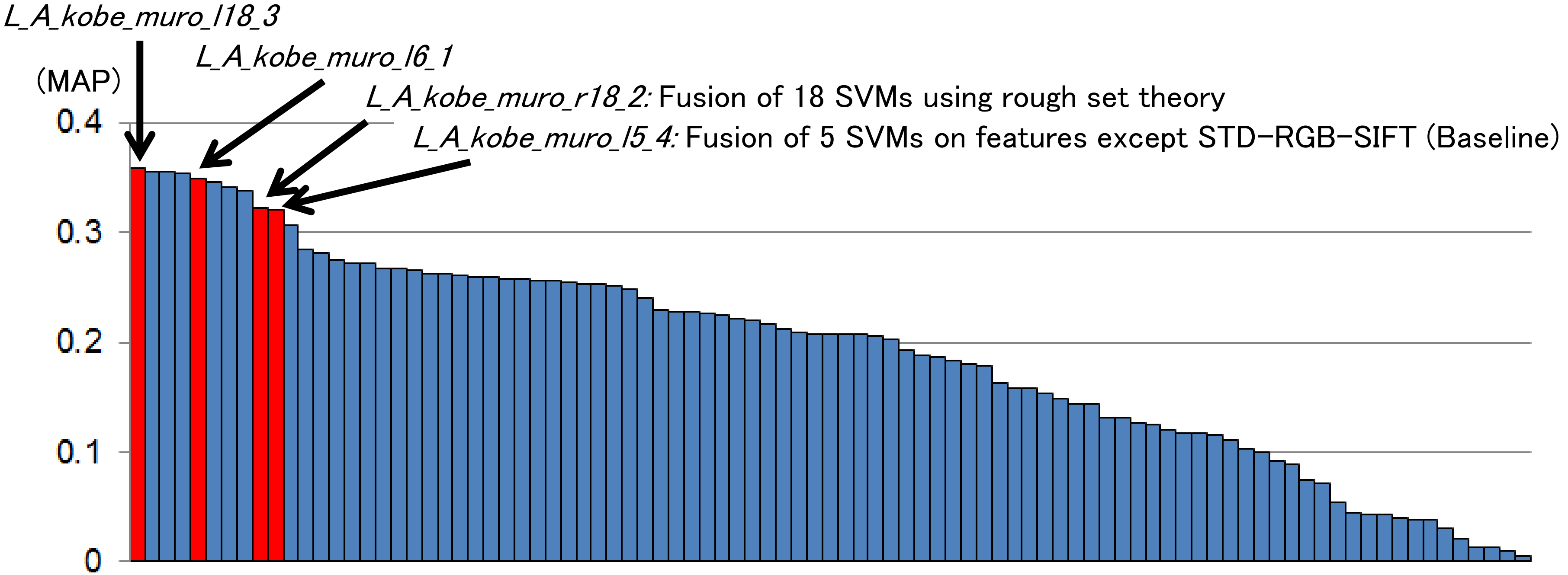
L_A_kobe_muro_l18_3: Weighted linear fusion of 18 SVMs based on bagging

– For each feature, three SVMs are built using different sets of 30,000 training examples (randomly selected three-quarter of positives, and randomly selected negatives)

– Three SVMs on each feature are equally weighted using the weight obtained in *L_A_kobe_muro_l6_1*.

The highest MAP (0.358) in SIN light task is achieved!

Much more improvement may be achieved using a more sophisticated fusion method.



Conclusion and Future Works

Fast and exact processing of large-scale video data based on matrix operation

1. Fast SVM training/test based on batch computation of kernel values
2. Fast spatially-temporally dense feature extraction based on batch computation of probability densities for multivariate normal distributions
3. Bagging to cover the diversity of a concept's appearances, by building many detectors with different sets of training examples

The efficiency or effectiveness of each approach has been confirmed.
→ *We achieved the highest MAP in TRECVID 2012 Semantic Indexing (light)!*

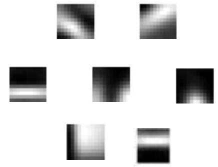
Future works

1. Development of a fast feature descriptor extraction method

(Raw shot)

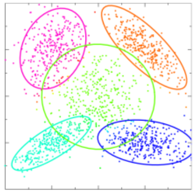
Slow

(Feature descriptors)



Fast
(batch computation)

(GMM representation)



→ Locality sensitive hashing: Feature descriptor extraction is skipped for regions, which are very similar to regions in the previous frame

2. Development of a sophisticated fusion method

Thank you!

Acknowledgement

We greatly appreciate the useful information from Mr. Inoue and Prof. Shinoda at Tokyo Institute of Technology