Using colour features for video-based tracking of people in a multi-camera environment

Mathew Price, Fred Nicolls, Gerhard de Jager

Department of Electrical Engineering
University of Cape Town
Rondebosch 7701, South Africa
mathew@dip.ee.uct.ac.za

Abstract

This paper provides a brief overview of ongoing work towards the realisation of a multi-camera video-based person tracking system. The developing system is orientated towards a distributed architecture for sharing visual information between multiple cameras. A hierarchical method for modelling people in scenes using colour features is proposed, however, future development will also look at including other useful measures such as texture. The method uses a kernel-based classification implementation to track a base set of visual components which can then be grouped to form high-level object descriptors. Some results are shown for a Kernel Adatron implementation of the classifier.

1. Introduction

Computer aided surveillance has been a developing area in computer vision circles for some time. With the increase in computer processing speeds and memory sizes, much past research is starting to have useful applications now that near real-time implementations are realisable.

This project has been primarily aimed at the global problem of tracking several people, simultaneously, through a multi-camera environment. Thus, in addition to dealing with single-view occlusions and scene changes, the system needs to be able to detect object movement between multiple camera views, which may, or may not, be overlapped. The problem faced is one that requires a system capable of coordinating data among several distributed resources in a manner that can be tied in with current computer vision methods. Additionally, this paper focuses on the use of colour as the primary discriminator between targets. In many situations colour is a powerful descriptor and can often reduce the complexity of a recognition problem immensely.

In terms of the generalised surveillance tracking class, a number of alternative approaches have been explored. The continual innovation in this area is attributed to the quest to find the ultimate system which is both robust and capable of real-time operation.

A well known colour tracking method involves the use of the mean-shift method [3]. This approach uses a histogram-based search which iteratively finds the most similar target candidate using a histogram similarity metric based on the Bhattacharyya coefficient. Its primary advantage is its low computational complexity and near real-time operation. However in the long-term, histogram modelling methods have been known to fail (even with adaptation). Also, the scaling of models for multiple objects and camera views is awkward.

Motion estimation techniques such as Kalman [15] and particle filters [7] approach the tracking problem from a stochastic point of view. If objects are parameterised by their physical interactions with an environment, and these parameters are assumed to follow a basic motion model corrupted by some noise, then predictive estimates can be used to track each target's position in the context of its model. An advantage of this formulation is that it intuitively facilitates integration with 3-D based information. Both Kalman and condensation trackers have been applied to solving the multi-view correspondence tracking problem [10, 11, 13, 16]. However, performance is dependent on the number of objects and views in the scene as well as the initialisation process. In fact in the event of failure, these systems are often unable to recover due to the complexity of system re-initialisation.

Simpler approaches such as blob tracking using segmented masks combined with complex rule sets have also been attempted [8] with some good short-term results. Solving the problem of computational complexity within high dimensional parameter spaces, inconsistency of multiple camera hardware, and the lack of a standardised framework have kept tracking at the forefront of computer vision research.

The system proposed here approaches the tracking problem from a classification standpoint. Appearance models of people are created based on colour features and then used by a neural network to detect their presence in subsequent frames. This one-shot style of tracking has the advantage of being able to deal with occluding objects, movement between multiple camera views and asynchronous video feeds. In addition, the framework is extremely flexible, allowing optional integration with a variety of information such as camera pose, geometric features and motion tendencies. Further, since a classification-based system does not suffer from cumulative measurement errors the proposed system can be used in conjunction with popular estimation-based trackers, thus solving the re-initialisation problem.

The following sections describe a distributed classification system capable of tracking colour components belonging to moving objects and persons. Operation proceeds by breaking a scene into unimodal colour components which are then classified into temporary object classes using an online kernel adatron. The final stage involves refining the allocation of classified colours to their respective objects based on group and spatial dependencies, though this stage has not been completely refined as yet.

2. System Overview

A basic premise of the system is that data should be hierarchically divided so that different levels of information can be processed appropriately (Figure 1). In this application, our lowest data level is the video input from the camera network.

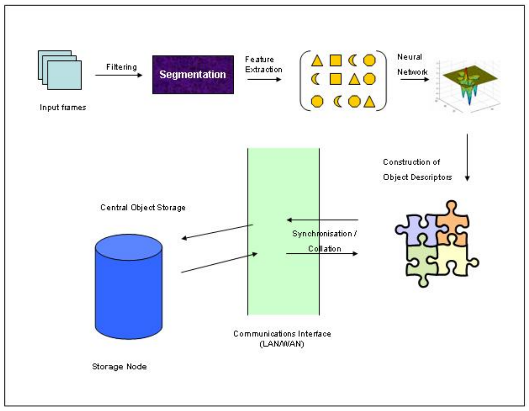

Figure 1: *System Data Flow Diagram.*

The first level of processing involves noise filtering, colour transformations and motion segmentation (separating moving foreground objects from the background image). Following this, the data enters the low-level processing level, where visual feature detection takes place (i.e. colour). These visual features are extracted using colour segmentation, and tracked using a neural network classification technique.

Once reliable visual features are available, construction of high-level descriptors of foreground objects is possible. These descriptors consist of combinations of the low-level visual features and can then be shared through the distributed processing system.

3. Feature Extraction

Extracting good features is probably the most critical task. While it is relatively simple to take a variety of measurements from images, it is imperative that one ensures that the feature measurements are both repeatable and comparable (i.e. the feature space is consistent and uniform).

Since the thrust of the project is to exploit the colour relationships of targets, a reliable chromatic measurement system is required. It was found that a convenient method for extracting colour, was to use CIE L*a*b* colour co-ordinates which preserve perceptual uniformity. The feature vectors thus appear as cartesian co-ordinates in $\mathbb{R}^3$.

Uniformity in the colour space refers to the property that standard distance measures (i.e. Euclidean) are proportional to the change in perceived colour. Owing to the fact that our neural network uses radial basis functions for its kernels, this uniformity becomes very useful when discriminating between colour clusters, since a translation in feature space relates directly to a change in perceived colour. This gives the system a human-like ability to distinguish colours. It is of course also possible to create an arbitrary colour space aimed at maximising colour discrimination, however, this discrimination will only be as good as the training set, and may not necessarily preserve uniformity.

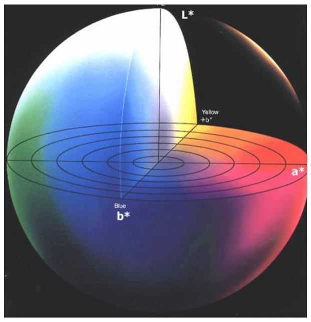

Figure 2: *CIE Lab Colour Space.*

The CIE L*a*b* space (Figure 2) consists of two colour axes and a luminance $L*$ or brightness axis. The $a*$ values range from red to green, while the $b*$ values describe blue to yellow. The configuration is similar to the HSV colour space, although the CIE L*a*b* space is spherical and the co-ordinates, being rectangular, allow the Euclidean distance between points to be relative to the difference in colour. A disadvantage is its computationally intensive transform, which requires an initial mapping to the CIE XYZ space.

Initially, during training, a motion segmented image is useful for isolating moving objects which need classification. However, once enough observations have been made, it is possible to switch back to classifying unsegmented images, which would decrease classification errors due to bad segmentation.

3.1. Colour Segmentation

Apart from motion segmentation, a colour segmentation scheme is required to generate a list of blobs and scene colours. This allows the object colours to be represented as features while retaining the spatial information (unlike histogram methods).

A batch colour segmentation method allows an efficient quantisation of the foreground into coloured components, leading to a compressed data representation which is suited to high-level processing. As colour segmentation applies to the training of object colour models, the process operates on the segmented foreground mask produced by motion segmentation.

Multiscale image processing techniques are hierarchical processes which use multiple resolutions of an image to perform analysis. These can be approached as top-down (quad-tree decomposition) or bottom-up processes (image pyramids). In the latter methods, image levels are constructed by downsampling the image by a series of filters — Gaussian in this case — which provides a smoother output.

The OpenCV library [6] provides a well optimised implementation based on the algorithm proposed in [2]. The input image (largest) can be thought of as the base of the pyramid. Each consecutive level is then built upwards, downsampling by a factor of two at each stage, until a specified maximum level is reached (commonly between 3 and 5). Two thresholds, T_1 and T_2, determine the nature of the segmentation, and linking is performed in two stages illustrated in Figure 3.

1. A link between a pixel $p_{x,y}$ on level L and its candidate father $p'_{x',y'}$ on level $L+1$ is established if: $\text{dist}(c(p_{x,y}, L), c(p'_{x',y'}, L+1)) \leq T_1$.

2. Connected components A and B are then clustered together if: $\text{dist}(c(A), c(B)) \leq T_2$.

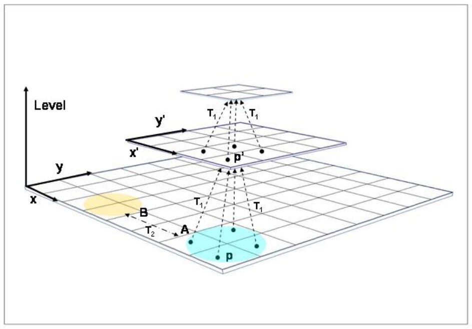

Figure 3: *Pyramid segmentation process.*

The function $dist$ is the Euclidean distance between local property functions c for each pixel. The local property c is the type of measurement associated with a pixel, which in this case is simply the intensity of the pixel. Therefore, the distance calculated by the $dist$ function is effectively the difference in pixel intensities $|i(x, y) - i(x', y')|$.

Parameter tuning, which is dealt with in [9], was found to be fairly inconsequential. Unlike [9], the primary goal here is to reduce the mass of pixel data to a manageable number of classes (approximately 5% of the image pixels). Therefore exact parameters for maximising the quantisation are not required. In fact, a larger list of small regions which more accurately describe the original pixel colours is preferred. In general, setting $T_1 = 30$ and $T_2 = 10$ provided equitable results.

The output of the pyramid segmentation method (Figure 4) is then processed into a list of image regions (connected components), each representing a unimodal colour cluster in the image. The CIE L*a*b* co-ordinates of these clusters are then used as features to train a neural network which can then classify future image regions into a basic colour set. Several clusters in this set can then be combined, by exploiting neighbourhood information, to form a high-level person/object descriptors.

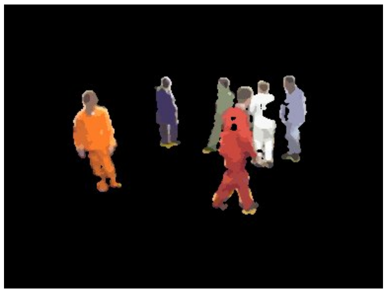

Figure 4: *Pyramid Colour Segmentation.*

4. Neural Network

As described previously, the base function of the system consists of being able to correctly identify the set of target colour clusters between observations. The classification process entails providing a probability comparison between the set of target colours and the visible colour components in the latest image. Since the feature data is generated by selecting Gaussian colour clusters, a Gaussian kernel-based probabilistic network seemed the logical choice.

4.1. Kernel Adatron

Several classification techniques could be easily applied to this model, and one could argue that a simple K-Nearest Neighbour classifier could easily do the job. The selection of the Kernel Adatron in particular is based on the overall needs of the system. Firstly, each model needs to be easily structured for global synchronisation with network storage nodes. Secondly, a degree of compactness is required both for speed in classification/updates and in storage. Finally, since observed data may often fall near the edge of the cluster group misclassification could cause invalid updates to be made, thus causing the system reliability to degrade.

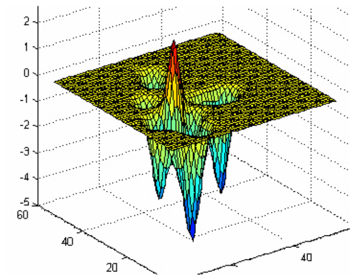

Figure 5: *Kernel Adatron example for 2-dimension features*

The kernel adatron (KA) is a simple implementation of a support vector machine, which allows maximisation of the margin in the feature space, thus forming a non-linear decision boundary in the input space [14]. Effectively, this provides an optimal decision boundary between classes. In the implementation, two target classes are represented by positive and negative unit Gaussians respectively. This allows a classification process to simply sum the Gaussian weights and classify between the two classes by the sign of the resulting probability (+1; -1). Figure 5 shows an example of a constructed KA. The positive peak identifies the current target class, while the negative peaks show other classes. Regions where the graph is zero are unclassified.

The largest difference between the KA method as opposed to the PNN (Probabilistic Neural Network) is that, during training, the KA assigns and updates a set of importance weights (α) which identify how close each kernel is to the decision boundary (support vectors). The primary advantage to our application here is that kernels with very small alpha values are not important to decision making and can therefore be discarded for classification. This results in a pseudo-compression scheme for the feature vectors, which constrains the models to a finite size and therefore limits the load of the network. Effectively, the compression works in a similar manner to an adaptive averaging filter, with the advantage that previous features are not necessarily averaged by time but rather by importance to classification.

4.1.1. Implementation

One of the features of the Kernel Adatron is that it is extremely easy to implement since it relies on only a few matrix operations. This implementation is extended from course literature by Green J. (Soft Computing 1999, UCT Press) which summarises theory presented in [14].

X_T — (m x n) matrix of m input features (rows) with n dimensions (cols).
T — (m x n) matrix of target values for n classes, each having m features.
A — (m x n) matrix of alpha values for n classes, each having m features.

The X_T matrix is a maintained set of features which determine the characteristics of each kernel. T assigns the target values for each feature vector. Columns in the T matrix represent each class, while row values identify whether the respective feature belongs (+1) or does not belong (-1) to the class. A typical target column will contain at least one $+1$ value. The A matrix has identical dimensions (and representative form) to the T matrix, but instead maintains the α weights (decision importance) for each feature (row) in the X_T matrix. If only one feature per class is stipulated, the A and T matrices become square.

4.1.2. Training:

For brevity we notate the squared Euclidean distance between input vectors x_1 and x_2 by $d^2(x_1, x_2)$. In the case where the input vectors consist of multiple rows, this operation returns a distance matrix describing the distance between each combination of inputs. For further information refer to [12].

Training proceeds in the following manner (referenced equations are defined below):

1. Calculate squared Euclidean distance matrix (Eqn 1)
2. Activate Gaussian kernels (Eqn 2)

   While less than i iterations **AND** $m < 0.99$ **DO:**

3. Calculate signed and weighted kernels (Eqn 3)
4. Update α values based on learning rate η (Eqn 4)
5. Calculate margin - distance from decision boundary to nearest point (Eqn 5).

$$D = d^2(X_T, X_T) \tag{1}$$
$$K = e^{-1/(2\sigma).D} \tag{2}$$

While less than i iterations **AND** $m < 0.99$

$$Z = K.A.T \tag{3}$$
$$\delta A = \eta(1 - Z.T) \tag{4}$$
$$m = 0.5\Big(min(Z^+) - max(Z^-)\Big) \tag{5}$$

4.1.3. Classification:

The classification procedure is identical to training steps 1 through 3 except that the distance matrix is now calculated between X and X_T, where X are the features to be classified.

4.1.4. Update:

Updating is simply a selection process where features classified positively in a class and meeting a similarity threshold are updated. In addition, features which were unclassified (sum of kernels = 0) and are significantly different from all other classes are appended to the X_T matrix. This addition obviously includes initialised updates to the A and T matrices as well.

5. Object Models

Once a reliable set of classified colour components is available, more complex grouping of the components can be used to create high-level visual object/person descriptors. Since several objects can share similar colours and since not all colours may always be visible, these descriptors cannot be represented by a combined feature vector. The objective is to group colour features that tend to cluster both spatially and temporally. These groupings should then correspond with the set of tracked targets.

One idea is to use a voting system where each object votes for a component based on basic region measurements, such as aspect ratio and relative magnitude, as well as its neighbouring colour relationships. Superposition of several separate decisions combined with the component's classification probability can then provide a comparative measure, indicating to which object model a colour component belongs.

6. Results

Colour component tracking by means of kernel-based classification has proved to be a highly effective method for locating characteristic regions of both people and objects. The current implementation is able to distinguish trained objects and persons with fair robustness to moderate lighting differences. The primary goal of this section of work is to assess the possibility and usefulness of distinguishing people or objects in a surveillance scene based on their colour. This by no means implies that colour should be the only descriptor, but allows its limitations and uses to be realised. Unfortunately, since the implementation forms part of on-going work, no quantitative comparisons could be drawn with other systems owing to the large deviation from the conventional tracking approach (classification-based scheme).

Some example images of classifying various trained targets are shown below. In general the actual colour classification phase performs extremely well with a correct colour matching rate of between 80 and 90%. Errors experienced are primarily caused by CCD saturation due to very bright lighting or dark areas. Figure 6 shows two examples where a shirt and a box have been selected for classification. Boxes represent a successfully classified colour component belonging to a trained model.

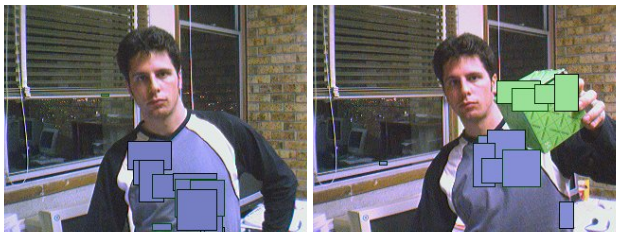

Figure 6: *Classification using unsegmented scene*

As seen in the right-hand image, errors can occur in classification when the target colour is visible in more than one place. Further work has been able to compensate for this by attaching importance weighting to targets based on their size and commonality throughout the scene. It is also evident that not all areas of the trained colour model have been detected. This is simply a matter of tuning the size of the radial basis function and has also been improved in further implementations.

Currently several camera servers allow images to be transmitted to a processing client over a 100 Mbps LAN during testing. Input frame rate varies between 5 and 15 fps depending on network load. Processing involving feature extraction and classification operates at an average of approximately 4 fps on a 320 x 240 unsegmented colour image processed on a Pentium III 450 MHz. It is envisioned that a more optimised implementation will lead to faster frame rates in the future.

Improved performance can be attained by using a motion segmented image as the input (Figure 7 shows an example). This reduces the processing required during pyramid colour segmentation (fewer pixels to process) and reduces the number of classifications necessary. In general, unsegmented input is only useful for scenes where continual environmental changes make segmentation unfeasible (e.g. reflective materials). On the other hand, the flexibility of the system to operate on either type of input may prove useful in applications such as tracking using a Pan Tilt Zoom or mobile camera.

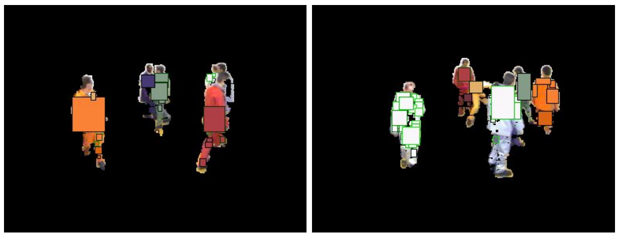

Figure 7: *Classification using motion segmented scene*

6.1. Discussion

A key issue, which affects the overall system, is colour consistency between different cameras. Since the entire goal of the system is to be able to correctly identify targets between multiple views, this can be exceedingly complex when colour measurements from two cameras are decidedly different for the same object. In addition, not only will each camera present a varied representation of the same scene, but it is unlikely for them to be orientated identically, and therefore the environment's lighting configuration could also affect the way the camera perceives objects.

Testing has shown that certain vibrant colours are usually detected between views, even without correction. In the general case, however, a calibration scheme will be needed to ensure that colour features match between views. Several automatic methods have been reviewed which are aimed at providing colour constancy [1], based on certain assumptions (e.g. Lambertian reflectance), and independence from device gamma [4]. Most of these methods are aimed at offline processing for image retrieval systems and are therefore not well suited to real-time multi-camera environments.

Austermeier et al. [5] have shown a promising method, which is being explored, for performing an unsupervised, target-based calibration scheme for normalising illumination changes. Their tests showed that a cloud of RGB pixels (plotted by omitting their spatial image placement) preserves its topology when subjected to a change in illumination. Furthermore, if the clouds of the original and resulting images are each quantised by a set of Self-Organising Map (SOM) prototypes, pixel colour can be corrected by simply translating its prototype between the two maps. SOMs have the useful feature of being able to quantise data into a set of prototypes, while at the same time preserving topological relationships between neighbouring neurons.

To clarify, usually the main usage of SOMs is for dimension reduction of feature data. However, in this application it is simply used as a 3-D data-fitting method. Another common practice is to use a 2-D neuron grid for the SOM. The main reason is because its distance matrix (depicting clustering information) is best understood in planar form. Once again, since this scheme is using the SOM's fitting ability, it is necessary to use an exact representation of the data, and thus the map is created as a 3-D lattice.

7. Conclusions

In conclusion, a realisable multi-camera implementation based on a distributed paradigm should prove to be a solid foundation for data exchange and synchronisation.

Approaching the tracking problem from a classification viewpoint in order to complement conventional tracking methods may prove a viable way of attaining a more robust person tracking system.

Using CIE L*a*b* colour features allows good comparison in feature space. At present, pyramid segmentation shows the most promise in providing a breakdown of colour components both in terms of speed and quality.

A kernel-based classification approach was favoured since the features are already grouped in a unimodal fashion. The Kernel Adatron Support Vector Machine implementation is a simple method that ensures optimum classification while also shedding redundant features and allowing a compact class representation.

Although the present implementation can correctly identify targets between different cameras, the accuracy depends on lighting conditions and the perspective viewing angle of each camera. For this reason a more robust colour calibration method is needed to ensure colour constancy between multiple camera views.

Finally, a large benefit is that the system can classify targets without a motion segmented image, thereby extending its application to other areas including: identifying targets or locations with mobile cameras; and tracking targets with a Pan Tilt Zoom camera.

8. Acknowledgements

Many thanks to TSS Technology, De Beers and the NRF for their continued support of this research.

9. References

[1] Rizzi A., Gatta C., and Marini D. A new algorithm for unsupervised global and local color correction. *Pattern Recognition Letters*, 24(11):1663–1677, 2003.

[2] P. J. Burt, T. H. Hong, and A. Rosenfeld. Segmentation and estimation of image region properties through cooperative hierarchical computation. *IEEE Transactions On Systems, Man, and Cybernetics*, 11(12):802–809, 1981.

[3] Dorin Comaniciu, V. Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE CVPR*, 2000.

[4] Finlayson G. and Xu R. Illuminant and gamma comprehensive normalisation in log rgb space. *Pattern Recognition Letters*, 24(11):1679–1690, 2003.

[5] Austermeier H., Hartmann G., and Hilker R. Color-calibration of a robot vision system using self-organizing feature maps. *ICANN*, pages 257–262, 1996.

[6] Intel. *Open Source Computer Vision Library Reference Manual*. Intel Corporation, USA, 2001. [Online]. Available: http://www.sourceforge.net/projects/opencvlibrary/index.html [February 2004].

[7] Michael Isard and Andrew Blake. Condensation — conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[8] McKenna S. J., Jabri S., Duric Z., Rosenfeld A., and Wechsler H. Tracking groups of people. *Computer Vision and Image Understanding*, 80:42–56, 2000.

[9] A. Kosir and J. F. Tasic. Pyramid segmentation parameters estimation based on image total variation. In *Proceedings of IEEE conference EUROCON 2003*, 2003.

[10] B. Merven, F. Nicolls, and G. de Jager. Multi-camera person tracking using an extended kalman filter. In *Fifteenth Annual Symposium of the Pattern Recognition Association of South Africa*, 2003.

[11] A. Mittal and L. S. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo. In *Computer Vision - ECCV 2002, 7th European Conference on Computer Vision, Copenhagen, Denmark, May 28-31, 2002, Proceedings, Part I*, 2002.

[12] I. Nabney and C. Bishop. Netlab neural network software. [Online] Available: http://www.ncrg.aston.ac.uk/netlab, [November 2003].

[13] K. Nummiaro, E. Koller-Meier, T. Svoboda, D. Roth, and L. Van Gool. Color-based object tracking in multi-camera environments. In *25th Pattern Recognition Symposium, DAGM 2003*, 2003.

[14] Friess T., Cristianini N., and Campbell C. The kernel-adatron algorithm: a fast and simple learning procedure for support vector machines. In *15th International Conference on Machine Learning*, 1998.

[15] Greg Welch and Gary Bishop. An introduction to the Kalman filter. Technical Report TR 95-041, Department of Computer Science, University of North Carolina, 2002.

[16] Z. Yue, S. K. Zhou, and R. Chellappa. Robust two-camera tracking using homography. Accepted for 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004.