BAP Sparsing: A Novel Approach to MPEG-4 Body Animation Parameter Compression

Siddhartha Chattopadhyay, Suchendra M. Bhandarkar and Kang Li
Department of Computer Science,
The University of Georgia, Athens, GA 30602-7404, USA
siddh@cs.uga.edu suchi@cs.uga.edu kangli@cs.uga.edu

Abstract

The MPEG-4 standard includes support not only for natural video and audio, but also for synthetic graphics and sounds. In the MPEG-4 specifications, Body Animation Parameters (BAPs) and Body Definition Parameters (BDPs) allow virtual bodies and their animation to be compressed using a standard compression pipeline comprising of quantization, predictive encoding and arithmetic coding of these parameters. In this paper, we propose and implement a new stage within the standard prediction-based compression pipeline for the BAPs, termed as BAP sparsing. BAP sparsing compresses a complete block of motion data, consisting of an initial I-frame followed by subsequent P-frames, required for creating the animation. It exploits the inherent hierarchical structure of the human skeletal model to intelligently drop and modify the P-frames, while preserving animation quality. BAP sparsing is shown to result in superior compression of the BAP data, with negligible loss in the motion animation quality. It is also shown to result in a lower network throughput requirement and fewer CPU cycles on the client end to create the animation. The proposed method is particularly well suited for animation using BAP data.

1. Introduction

The MPEG-4 standard seeks efficient representation and encoding of audio and video, together with synthetically generated audiovisual information, such as virtual humans [1] [2]. A virtual human body model is animated using a stream of body animation parameters (BAPs). BAP encoding is particularly suited for low-bitrate transmission in dedicated interactive communications and broadcast environments. The BAPs control the various independent degrees of freedom in the skeletal model of the body to produce animation of the body parts. The BAP data enable real time transmission, rendering and manipulation of life-like visual scenes of the human body on a remote device without the need for transmission of the pictorial and video details of the human body in every frame. An overview of body encoding, using MPEG-4 and MPEG-4 compliant avatar control, is given in [4].

An important application of BAP based animation is the streaming of the BAP data across the Internet to clients, such that real time animation is possible. A practical use of such an animation is in intelligent next-generation distributed human-computer interaction. Issues pertaining to video streaming over the Internet have been well researched [9] [10] [12] [13], and a standard compression pipeline for reduced throughput has been established over time. Recent standards in MPEG-4 allow streaming of BAPs over the Internet. Compression of BAP data for efficient network transmission with low bit-rates can be represented by a pipeline of stages that yields a lossless compressed version of the original data [5]. A lossy adaptation of the existing BAP compression technique includes frame dropping, which results in loss of motion animation quality. A major drawback of the conventional lossy compression pipeline that uses frame dropping is the lack of intelligent, data-and model-dependent compression; i.e. the inherent structure of the underlying model is not utilized. Hence, it is desirable to have a more sophisticated frame dropping algorithm that exploits the structure of the underlying model.

In this paper, we introduce a new stage within the MPEG-4 BAP compression pipeline, termed as *BAP sparsing*, which drops and modifies frames intelligently by exploiting the hierarchical structure of the human skeletal model. The proposed method results in a higher compression ratio compared to the existing MPEG-4 BAP compression pipeline, and enables control of the motion animation quality via a single tunable parameter.

Proceedings of the 2005 Systems Communications (ICW'05)
0-7695-2422-2/05 $20.00 © 2005 IEEE

In order to evaluate the proposed method, we have used the compression algorithm offline on MPEG-4 BAP motion files. The motion is represented by a matrix consisting of a single I-frame, followed by a sequence of P-frames. Our compression algorithm results in a modified version of the matrix with many elements of the matrix reduced to zero (sparsing). The motion resulting from the new matrix is observed to be minimally distorted compared to the motion derived from the original matrix (Section 3). The sparse matrix is the offline emulation of P-frame dropping, and leads to an improved compression ratio, resulting in a lower network throughput requirement for streaming the motion data. The performance comparison is done essentially between the animations resulting from the use of the sparse matrix and the non-sparse matrix.

In the sections to follow, we first give a brief description of the current MPEG-4 BAP compression pipeline, followed by the description of the proposed *BAP sparsing* stage. In the results section, we provide a quantitative evaluation of the proposed method in terms of motion quality, network throughput requirement, compression ratio and CPU power consumption, followed by a qualitative comparison of the proposed technique with the conventional frame dropping technique.

2. MPEG-4 BAP Compression

The Face and Body Animation (FBA) object is a collection of nodes in a scene graph, which are animated using two separate FBA object bit-streams, called BIFS (BInary Format for Scenes). The first bitstream contains instances of Body Definition Parameters (BDPs) in addition to Facial Definition Parameters (FDPs), and the second bitstream contains Body Facial Animation Parameters (FAPs) [3].

The BAPs are compressed for efficiency using a standard compression pipeline comprising of quantization taking into account the physical constraints of the joints, representation using prediction errors and finally arithmetic coding for generic bit-level compression. To achieve further compression, frame dropping is used to reduce the number of transmitted frames, consequently reducing the network throughput requirement and hence the bit-rate. Assuming no packet loss, the BAP stream consists of a single I-Frame followed by all the predicted P-Frames necessary to render the complete animation. For a new animation sequence, another set of data, consisting of an I-frame followed by a long sequence of P-frames is streamed.

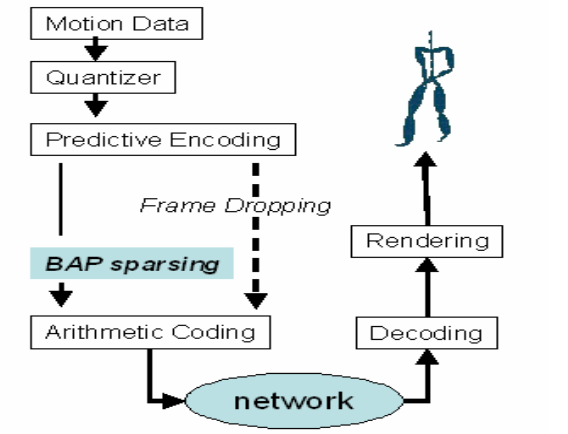

Figure 1: The standard compression and decompression pipeline for MPEG-4 BAPs, enhanced with a novel alternative stage termed as *BAP sparsing*.

2.1. Error Sparsing Stage in the Compression Pipeline

We introduce a *BAP (error) sparsing* stage between the error encoding and arithmetic coding stages of the standard MPEG-4 BAP compression pipeline (**Figure 1**). The basic intuition underlying *BAP sparsing* is to allow the joint corresponding to the BAP parameter to be *frozen*, for a fraction of a second, and then *released* all of a sudden. However, once released, the joint corresponding to the BAP parameter will have greater cumulative displacement to make up for the lost displacements when frozen. If performed judiciously, the *freeze-release* operation should be imperceptible to the human eye.

To implement this idea, we consider each BAP parameter, and drop consecutive P-frames, and accumulate the values of the dropped P-frames until the cumulative value exceeds a predefined *threshold*. At this point, we replace the current P-frame value by the cumulative value, and reset (to zero) the variable corresponding to the cumulative value. The same process is repeated for succeeding frames until all the P-frames have been scanned once. In the remainder of this section, we detail the sparsing algorithm and the computation of the threshold value.

Without loss of generality, it is assumed that the BAP P-frame encoding error data is represented by an n x m matrix $\mathbf{X}_{n \times m}$, where n is the number of frames of the animation, and m is the number of BAP parameters to be used for defining a pose of the model (the maximum value for $m = 296$ as defined in the MPEG-4 standard). Initially, the mean $\mathbf{m_i}$ and standard deviation $\mathbf{s_i}$ of each column (parameter) i is computed. For each instance of each BAP parameter i that passes through

```
for k ← 1 to m
   S_k = X_1,k
for i ← 2 to n
   for k ← 1 to m
      S_k ← S_k + G_i-1, k
      if ((|S_k - m_k |/ S_k) > T_k)
         B_i, k = S_k
         S_k  = X_i, k
      else
         B_i, k = 0
for i ← 2 to n
   for k ← 1 to m
      X_i-1,k = B_i, k
```

Figure 2: The algorithm for *BAP Sparsing*, assuming the P-Frames are in a matrix format $\mathbf{X}_{n \times m}$. $\mathbf{G}_{i,k} = \mathbf{X}_{i,k} - \mathbf{X}_{i-1,k}$. **B** is a temporary matrix, whose values are finally assigned to **X**.

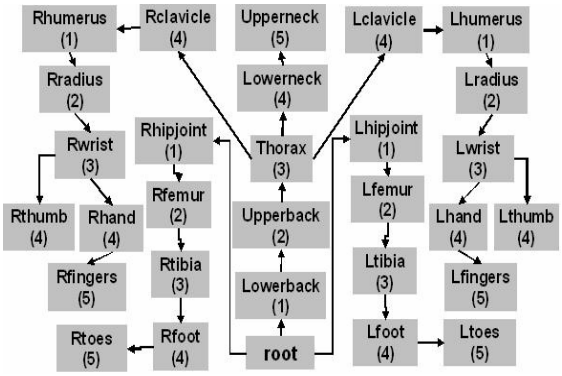

Figure 3: An example of hierarchical structure of a human consisting of 31 nodes, with a total of 62 degrees of freedom of motion (rotational and translational). For convenience, the root node is drawn at the bottom.

the compression pipeline, a variable $\mathbf{S_i}$ is used to accumulate the difference of predictive encoding errors, in successive frames. The current encoding error is set to zero (*sparsing stage*), until the normalized cumulative encoding error ($|\mathbf{S_k} - \mathbf{m_k}| / \mathbf{s_k}$) exceeds a predefined threshold value $\mathbf{T_i}$. The value of the threshold $\mathbf{T_i}$ is determined using information derived from the hierarchical skeletal model of the human, for which the BAPs are being used. The error sparsing algorithm is given in **Figure 2**. A detailed discussion on the determination of threshold $\mathbf{T_i}$ is presented in the next section.

The centered and normalized value of the cumulative predictive encoding ($|\mathbf{S_k} - \mathbf{m_k}| / \mathbf{s_k}$) is used in order to determine the value of the threshold $\mathbf{T_i}$ irrespective of the mean and spread of the data in the columns of **X.** This stage renders the algorithm to be more general, spanning a wide range of motions, and enables the setting of a default value of the control parameter.

2.2. Determination of the Threshold T_i

We compute the threshold value for each BAP parameter by exploiting the hierarchical structure of the underlying human skeletal model. The basic observation that we have utilized is the fact that the threshold value for a particular BAP parameter should depend on the position of the corresponding body joint in the human skeletal model. For example, consider the hierarchical human skeletal model given in **Figure 3**. Any joint higher in the hierarchy is allowed less angular error compared to joints lower down in the hierarchy. For example, an angular error at *Rhipjoint* displaces the joint *Rfoot* more than the same angular error at the joint *Rtibia*, which is lower down in the hierarchy.

We have represented the hierarchical significance of the various joints in the human body by a level number, as shown in the parentheses in **Figure 3**. The threshold $\mathbf{T_i}$ for a joint *i*, represented by column *i* in matrix $\mathbf{X}_{n \times m}$, should be such that the joints higher in the hierarchy, with a smaller level value, should be allowed *less angular error* due to sparsing of the predictor error, when compared to joints with a larger level value. This ensures that the *displacement error* induced by the sparsing operation is small. For a column *i* of $\mathbf{X}_{n \times m}$, with level L_i, the corresponding threshold $\mathbf{T_i}$ is given by

$$\mathbf{T_i} = \boldsymbol{K} L_i \qquad (1)$$

where $\boldsymbol{K}$ is the *quality control parameter* (QCP). Typically, the value of $\boldsymbol{K}$ is chosen such that the maximum threshold value is less than 1.

3. Results and Discussions

We present the results and discussions in two subsections. The first subsection presents a quantitative analysis of the tradeoff obtained between the quality of motion animation, degree of compression achieved, reduction in power consumption (measured by the number of CPU cycles) and the network throughput requirement. The second subsection presents a qualitative comparison of the proposed technique with the conventional frame dropping technique.

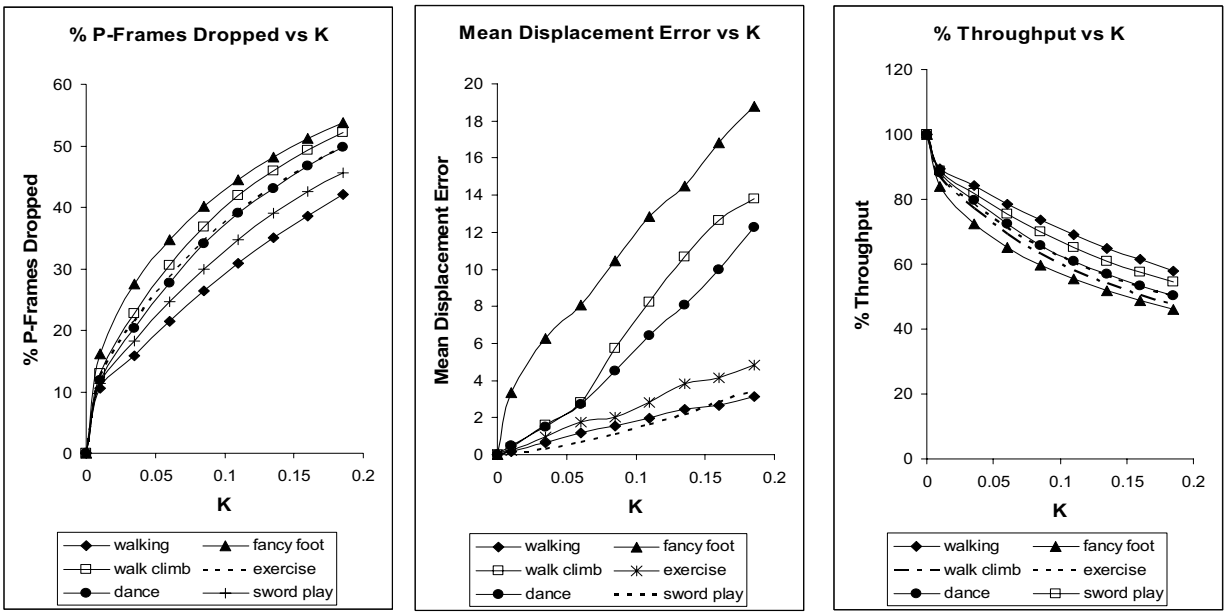

Figure 4: (**Left**) Percent P-Frames dropped vs Quality Control Parameter **K** (**Center**) Mean Displacement Error vs **K** (**Right**) Network throughput requirement vs **K**. Walking and exercise yield similar throughputs, hence the lines are superimposed.

3.1 Animation quality, network throughput requirement and power consumption

We have represented each motion matrix as a quantized matrix of the P-frames, and have tested our algorithm for different motions of varied duration and complexity. We present our results on the following representative motion examples:

- Walking: Simple periodic motions.
- Sword Play: Simple non-periodic motions.
- Walking-Climbing: Complex but slow motions.
- Exercises: Motions with multiple complexities.
- Dancing: Slow, but ill-correlated motions.
- Fancy Footwork: Fast, erratic and extremely ill-correlated motions, where the subject rolls on the floor and performs complex dance motions very rapidly.

We have observed the dependence of the percentage of P-Frames dropped, the mean displacement error and the percentage network throughput requirement for the above mentioned motion examples as a function of the quality control parameter **K** (**Figure 4**). The percentage network throughput requirement is simply the fraction of the original number of bytes to be transferred over the network. The mean displacement error is the cumulative displacement error induced by sparsing, normalized with respect to the number of frames in the motion sequence.

The mean displacement error depends on the nature of the motion. As shown in **Figure 4**, complex motions such as a combination of motions (walking and climbing) and erratic motions (fancy footwork) yield more noise/distortion in the motion data, compared to regular motions such as walking. The percentage network throughput requirement is greater for regular motions such as walking, whereas it is less for erratic and sudden motions such as fancy footwork. Although this seems counter-intuitive, it makes sense when the mean displacement error is also considered; for the same value of **K**, the animated motion results in different mean displacement errors depending on the type of motion. This means that the quality control parameter **K** is not an absolute universal measure of motion animation quality; the more erratic or complex the motion, the greater is the mean displacement error for a given value of **K**. Ideally, this should not be the case. However, at this juncture, it should also be noted that **K**, though not a perfect and universal quality control parameter, is adequate for most types of motions encountered in practice.

Since the motion animation quality, throughput requirement, and percentage of P-frame BAPs dropped are all dependent on the quality control parameter **K**, it is desirable to have a default, "safe" value of **K**, which guarantees an acceptable level of displacement error that is not obviously perceptible. Additional experiments and empirical studies with human observers have shown that choosing **K** = **0.1**, and

Table1

Motion	No. of Frames	Arithmetic Coding (KB)	BAP Sparsing + Arithmetic Coding (KB)	% reduction in file size	CPU Giga Cycles Encoding Original	CPU Giga Cycles Encoding Sparsed	CPU Giga Cycles Decoding Original	CPU Giga Cycles Decoding Sparsed	% reduc-tion in CPU Cycles
Walking	343	83	63	75.9%	1.341	1.031	0.928	0.756	81.5%
Dance	434	104	73	70.2%	1.684	1.169	1.169	0.859	73.5%
Sword Play	2251	515	386	75.0%	8.491	6.394	5.981	4.503	75.3%
Fancy Foot	2555	586	448	76.5%	9.625	6.394	6.841	4.538	66.3%
Exercise	4653	1024	743	72.6%	16.981	12.272	11.928	8.697	72.9%
Walk, Climb	4839	1082	776	71.7%	17.864	12.684	13.166	8.972	68.1%

Table 1: Comparison of the degree of compression and number of CPU cycles required to encode and decode motion data, with simple arithmetic coding, and arithmetic coding after *BAP sparsing*. The BAP sparsing step prior to arithmetic coding yields superior results in both cases. QCP **K** = **0.1** for all the experiments.

hence ensuring that the normalized cumulative predictive error (P-Frame values) is less than 1, yields negligible motion distortion for all the examples we have studied.

Since the MPEG-4 standard uses arithmetic coding as the final stage for compression of the BAP data, we have implemented and tested the amount of compression for various motion data, having processed them through the enhanced pipeline incorporating *BAP sparsing*, as depicted in **Figure 1**. We have obtained a significant amount of compression after the final arithmetic coding stage, with no visual distortion of the motion, with **K** = 0.1. The test results for some of the representative motions are given in **Table 1**.

For a power constrained device, both encoding and decoding of motion data using arithmetic coding are critical sources of power consumption in the case of streaming motion data. To estimate power consumption, we computed the number of CPU cycles needed to encode and decode the motion data for the representative motion examples mentioned above. We used a 2.2 GHz Intel Celeron Processor with 128 KB L2 cache and 512 MB RAM for running the arithmetic coding and decoding algorithms. As shown in **Table 1**, our method requires fewer CPU cycles for both the encoding and decoding process, from which it is safe to infer that the proposed method saves power during both encoding and decoding of the streaming motion data.

3.2 Comparison of BAP Sparsing and Frame Dropping

Trading bandwidth and CPU resource consumption with data quality has been a common practice, especially for media streaming over the Internet. The MPEG standard has been designed with the ability of making data quality adaptations by simply performing frame dropping. There exists considerable published research literature on this topic [6] [7] [11]. When compared to data quality adaptation using traditional frame dropping adaptations, *BAP sparsing* can be considered to be a "smart" data dropping technique. Instead of skipping frames and simply discarding the motion data, *BAP sparsing* reduces the amount of transmitted data by simplification and aggregation of the motion data to approximate the same underlying motion. *BAP sparsing* can achieve this by taking advantage of the knowledge of the moving objects and their internal physical representation.

BAP sparsing has a significant advantage over the simple frame dropping technique, especially for animation video. For example, in the case of traditional MPEG random frame dropping, once a P-frame is dropped, all its subsequent frames within the same GOP and even some B-frames, transmitted before the dropped P-Frame that refer to the P-Frame, have to be skipped because of decoding failure. For a normal video clip, because of the relatively small GOP size [8], the quality degradation caused by P-frame dropping is constrained to lie within a very short time interval (determined by the GOP size). The video quality can recover to its original value when the first frame of the next GOP arrives. Unfortunately, the BAPs are organized as a single long GOP with a single I-frame at the beginning of the sequence followed by several P-frames. This makes traditional frame dropping unsuitable for data quality adaptation in the standard BAP compression pipeline, since the quality of BAP data is severely compromised. *BAP Sparsing*, on the other hand, understands the motion data and the associated skeletal model hierarchy, and keeps track of

the accumulated error so that P-frames can be skipped and modified without affecting future frames.

4. Conclusions and Future Work

A new stage, termed as *BAP sparsing*, has been proposed within the standard MPEG-4 BAP compression pipeline, as an alternative to the standard frame dropping technique used by many MPEG-4 compression algorithms. The resulting motion distortion is minimized by using smart P-frame dropping and P-Frame error recomputation, which exploit the hierarchical information obtained from the skeletal model of the human. The additional stage in the compression pipeline is required only on the server side. Thus, the client side interprets the result as a single I-frame followed by the P-frames necessary for the animation, without having to perform any additional computations.

Our results have shown significant improvements in compression ratio for all the motion examples that we have experimented with. The proposed method yields compressed motion data which requires much fewer CPU cycles, resulting in much less power consumption, compared to the standard compression technique, for both encoding and decoding. The quality of the motion animation and resulting network throughput requirement is controlled by a single quality control parameter **K**, whose default value is set to **0.1**. With the default value of **K**, it is possible to compress the motion video in an unsupervised manner while simultaneously ensuring acceptable quality of the resulting motion animation. The proposed method is generic, and can be used for H-Anim MPEG-4 characters of any structural complexity.

There is scope for future improvements in the proposed method. Although the current default value for the QCP **K** = 0.1 yields satisfactory results, it is desirable to obtain an optimal value for the QCP **K** via statistical analysis of the underlying motion data. A good metric for quantification of the quality of motion is needed, as the current measure based on mean displacement error does not map to any direct measure of the visual quality of motion animation.

References

[1] ISO/IEC 14496-1:1999, "Coding of Audio-Visual Objects", *Systems*, Amendment 1, December 1999.

[2] ISO/IEC 14496-2:1999, "Coding of Audio-Visual Objects", *Visual*, Amendment 1, December 1999.

[3] Capin T. K., Petajan, E., and Ostermann, J., "Efficient Modeling of Virtual Humans in MPEG-4", *Proc. ICME'2000*, New York, NY, July 2000.

[4] Capin, T. K., and Thalmann, D., "Controlling and Efficient Coding of MPEG-4 Compliant Avatars", *Proc. IWSNHC3DI'99*, Santorini, Greece, 1999.

[5] Capin, T., K., Petajan, E., and Ostermann, J. "Very Low Bitrate coding of virtual human animation in MPEG-4". *IEEE International Conference on Multimedia and Expo (ICME)*, New York, NY, Volume: 2, 2000.

[6] Rowe, L. A., Patel, K. D., Smith, B., C., and Liu, K., "MPEG Video in Software: Representation, Transmission, and Playback", *Proc. High Speed Networking and Multimedia Computing*, San Jose, CA, February 1994.

[7] Krasic, C., Walpole, J., and Feng, W., "Quality-Adaptive Media Streaming by Priority Drop", *Proc. 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2003)*, Monterey, California, June 2003.

[8] Feng, W., Choi, J., Feng, W., and Walpole, W., "Under the Plastic: A Quantitative Look at DVD Video Encoding and Its Impact on Video Modeling", *Proc. Packet Video*, Nantes, France, April, 2003.

[9] Wu, D., Hou, Y., Zhu, W., Zhang, Y., and Peha, J., "Streaming Video over the Internet: Approaches and directions", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 1, pp. 1-20, Feb. 2001.

[10] Kang, S. and Zakhor, A., "Packet Scheduling Algorithm for Wireless Video Streaming", *Proc. 12th Packet Video Workshop 2002*, Pittsburg, PA, Apr. 2002.

[11] Zhang, Z. L., Nelakuditi, S., Aggarwal, R., and Tsang, R. P., "Efficient Selective Frame Discard Algorithms for Stored Video Delivery across Resource Constrained Networks", *Proc. IEEE Infocom 99*, New York, Mar. 1999, pp. 472-479.

[12] Elsen, I., Hartung, F., Horn, U., Kampmann, M., and Peters, L., "Streaming Technology in 3g Mobile Communication Systems", *IEEE Computer*, vol. 34, no. 9, pp. 4652, September 2001.

[13] McNamee, D., Krasic, C., Li, K., Goel, A, Steere, D., and Walpole, J., "Control Challenges in Multi-Level Adaptive Video Streaming", *Proc. IEEE Conference on Decision and Control (CDC2000)*, Sydney, Australia, Dec, 2000.