Activity Detection in Conversational Sign Language Video for Mobile Telecommunication

Neva Cherniavsky, Richard E. Ladner, and Eve A. Riskin †
Department of Computer Science and Engineering, Box 352350
† Department of Electrical Engineering, Box 352500
University of Washington, Seattle, WA 98195
{nchernia, ladner, riskin}@cs.washington.edu

Abstract

The goal of the MobileASL project is to increase accessibility by making the mobile telecommunications network available to the signing Deaf community. Video cell phones enable Deaf users to communicate in their native language, American Sign Language (ASL). However, encoding and transmission of real-time video over cell phones is a power-intensive task that can quickly drain the battery.

By recognizing activity in the conversational video, we can drop the frame rate during less important segments without significantly harming intelligibility, thus reducing the computational burden. This recognition must take place from video in real-time on a cell phone processor, on users that wear no special clothing.

In this work, we quantify the power savings from dropping the frame rate during less important segments of the conversation. We then describe our technique for recognition, which uses simple features we obtain "for free" from the encoder. We take advantage of the conversational aspect of the video by using features from both sides of the conversation. We show that our technique results in high levels of recognition compared to a baseline method.

1. Introduction

Mobile phones are rapidly becoming ubiquitous, with over 2.68 billion mobile phone subscribers worldwide [16]. In the United States, federal law has long mandated accessibility to the telephone network for the Deaf through subsidized telephone typewriters (TTY) and, more recently, video relay service. However, there is no equivalent service for access to mobile telecommunications. The MobileASL project [5, 9] aims to expand accessibility for Deaf people by compressing sign language video to enable mobile phone communication. With today's bandwidth limitations [12] and the low processing power available on phones, two-way real-time mobile sign language communication is not feasible using current video compression technology in the United States.

However, in the same way that there are characteristics unique to speech that allow speech to be compressed more than standard audio, sign language has distinct features that should enable better compression than is typical for video. In this work, we focus on recognizing activity in sign language video in order to make adjustments that might increase or maintain intelligibility while decreasing cost. Cost can be measured in several ways: dollar value, if users are expected to pay based on how much data they transmit; processing power, so that real-time compression may be achieved on a standard phone; and battery life, since a short-lived phone is not very useful. One way to save data transmission and processor cycles while minimally affecting intelligibility is to lower the frame rate on the basis of the activity in the video [8]. Because conversation involves turn-taking (times when one person is signing while the other is not), we may save power as well as bit rate by lowering the frame rate during times of not signing, or "just listening" (see Figure 1).

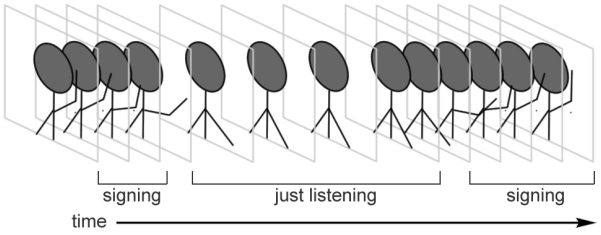


Figure 1. From left to right: a sufficient video frame rate is chosen when the signer is signing, the frame rate decreases when the signer is not signing (or just listening), and increases again when the signer begins signing.

Our goal is to recognize the activity from a video stream in real-time on a standard mobile telephone. Since we want

978-1-4244-2154-1/08/$25.00

to increase accessibility, we do not restrict our users to special equipment or clothing. We only have access to the current frame of the conversational video of the signers, plus a limited history of what came before.

To accomplish our task, we harness two important pieces: the information available "for free" from the encoder, and the fact that we have access to both sides of the conversation. The encoder we use is H.264, the state-of-the-art in video compression technology. H.264 works by finding motion vectors that describe how the current frame differs from the previous one. We use these, plus some additional features, as input to a support vector machine. We then improve our results by taking advantage of the two-way nature of our video. Using the features from both conversation streams does not add complexity and allows us to better recognize the activity taking place.

The paper is organized as follows. In the next section, we briefly discuss related work. In section 3, we demonstrate power savings on a mobile phone when encoding and transmitting frame rate-adjusted videos. In section 4, we describe our feature extraction and machine learning techniques for real-time activity recognition on mobile telephones. Section 5 contains our results, where we compare our method to a simple thresholding technique and show that we outperform it. We conclude in section 6.

2. Related Work

Sign language recognition is well-studied in the literature [18]. While related to our work, our goal is not translation or interpretation, but rather increasing accessibility by enabling mobile telecommunication for the signing Deaf community. We are constrained to on-line algorithms that are efficient enough to recognize activity in real-time on a mobile phone. Our users should be able to use our software on a standard phone, without wearing special clothing or using additional equipment.

We build on the work of Cherniavsky *et al*. [8]. They performed user studies with native signers to examine intelligibility at different frame rates for conversational sign language video. They also showed the feasibility of using machine learning to recognize different activity, though the baseline method sometimes outperformed the machine learning algorithm. They did not quantify the power use on the mobile phone, and the videos used for the machine learning were taken from a stationary camera on a tripod, with a dark background. Because of this, only one side of the conversation was available to train and test.

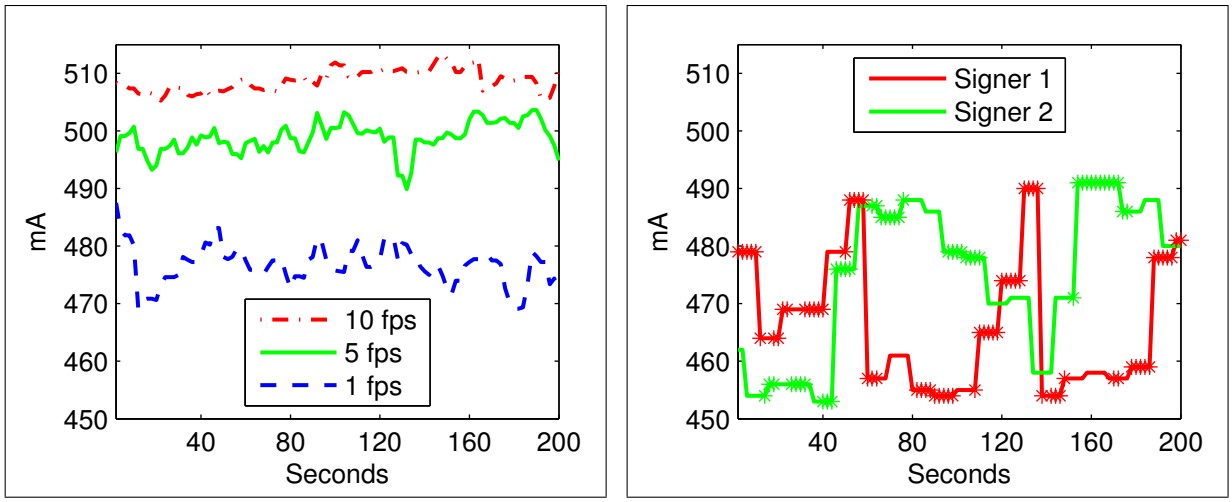
Most closely related to our work is vision-based sign language recognition. There are two main parts to any recognition task: feature extraction and machine learning. The goal of feature extraction is to find a reduced representation of the data that models the most salient properties of the raw signal. In vision-based recognition, the features must be extracted from video. If feature extraction is too slow to support a frame rate of 5 frames per second (fps), it is not real-time and thus not suitable to our purposes. This includes Huang *et al*. and Chen *et al*.'s Fourier descriptors to model hand shape [7, 13]; Cui and Weng's pixel intensity vector [11]; Huang and Jeng's active shape models [14]; and Tamura and Kawasaki's localization of the hands with respect to the body [22]. Though the time complexity was unreported, it is likely that Imagawa *et al*.'s principal component analysis of segmented hand images is not real-time [15]. Yang *et al*. also did not report on their time complexity, but their extraction of motion trajectories from successive frames uses multiple passes over the images to segment regions and thus is probably not real-time [25].

More promising for our purposes are the techniques that use the center of gravity (COG) of the hand and/or face. One way to easily pick out the hands from the video is to require the subjects to wear colored gloves. Assan and Grobel [3] and Bauer and Kraiss [4] use gloves with different colors for each finger, to make features easy to distinguish. Tanibata *et al*. use skin detection to find the hands, then calculate the COG of the hand region relative to face, the area of hand region, the number of protusions (i.e. fingers), and the direction of hand motion [23]. Kobayashi and Haruyama extract the head and the right hand using skin detection and use the relative distance between the two as their feature [17] . Starner *et al*. use skin detection to track the hands and extract the bounding ellipse and angle of least inertia of the skin blobs [21]. Following on these techniques, some of our features use the skin-detected areas of the video.

In the computer vision community, automatic activity analysis is an active topic of research. Though conversational sign language video has not been explored to our knowledge, shot change detection [20] (determining when a scene changes) and human motion analysis [24] are both widely studied. The first problem does not usually require on-line algorithms and solutions to the second problem are often not real-time and require processing power beyond the scope of a mobile phone. We use the original shot change detection method of differencing as a baseline to compare against.

3. Power Study

In order to quantify the power savings from dropping the frame rate during less important segments, we monitored the power use of MobileASL on a Sprint PPC 6700 at various frame rates. MobileASL normally encodes and transmits video from the cell phone camera. We modified it to read from an uncompressed video file and encode and transmit frames as though the frames were coming from the camera. We were thus able to test the power usage at different frame rates on realistic conversational video (described in Section 4). We used a publicly available power meter pro-



(a) Average power use over all videos

(b) Power use at 1fps for one conversation

Figure 2. Power study results.

gram [1] to sample the power usage at 2 second intervals. The minimum frame rate necessary for intelligible signing is 10 frames per second (fps), but rates as low as 1 fps are acceptable for the "just listening" portions of the video [8]. Thus, we measured the power usage at 10 fps, 5 fps, and 1 fps. Power is measured in milliamps (mA) and the baseline power usage, when running MobileASL but not encoding video, is 420 mA.

Figure 2 shows (a) the average power usage over all our videos and (b) the power usage of a two-sided conversation at 1 fps. On average, encoding and transmitting video at 10 fps requires 17.8% more power than at 5 fps, and 35.1% more power than at 1 fps. Figure 2(b) has stars at periods of signing for each signer. Note that as the two signers take turns in the conversation, the power usage spikes for the primary signer and declines for the person now "just listening." The spikes are due to the extra work required of the encoder to estimate the motion compensation for the extra motion during periods of signing, especially at low frame rates. In general the stars occur at the spikes in power usage, or as the power usage begins to increase. Thus, while we can gain power saving by dropping the frame rate during periods of not signing, it would be detrimental to the power savings, as well as the intelligibility, to drop the frame rate during any other time.

4. Activity Analysis

Our goal is to automatically detect from our video stream when the user is signing and not signing, so we can lower or raise the frame rate accordingly. We only have access to the information available to us from the video stream, and we must be able to determine the class of activity in real-time. We first apply feature extraction to the video stream, and then input those features to a support vector machine.

The conversational videos were recorded directly into raw YUV format from a web cam. Signers carried on a conversation at their natural pace over a web cam/wireless connection. Two pairs recorded two different conversations in different locations, for a total of eight videos. For each pair, one conversation took place in a "noisy" location, with lots of people walking around behind the signer, and one conversation took place in a "quiet" location with a stable background. We encode the videos with x264 [2], a publicly available implementation of H.264, the latest standard for video compression.

4.1. Features

The H.264 video encoder primarily works by encoding how the current frame differs from the previous one. If a video is encoded at a reasonable frame rate, it does not change much from one frame to the next. H.264 works by dividing a frame into 16×16 pixel macroblocks, then sending for each macroblock the location of the macroblock in the previous frame that it most closely resembles, plus some residual information (see Figure 3(a)). Macroblocks are in one of three categories: skip blocks, P blocks, or I blocks. Skip blocks indicate that this macroblock is exactly the same as in the previous frame; they are green in Figure 3(a). P blocks have motion vectors associated with them, and are sometimes subdivided into smaller portions. In Figure 3(a), these are orange and blue. I blocks are intra blocks, meaning that they get the pixel information from the current frame rather than the previous one. They usually indicate the most motion of all, because the encoder is unable to find a block in the previous frame that matches the current macroblock. These are the red blocks in Figure 3(a). Note that the I blocks are centered around the left

hand moving rapidly toward the right, while there are motion vectors associated with the slower motions of the right hand. The encoder chooses the cheapest of these in terms of bits and sends it.

For each frame, we obtain either motion vector information for each macroblock or an I block, which is an indication that the encoder gave up trying to find good motion vectors. Because our videos are conversational over mobile phones, the field of view is generally restricted to the head and upper torso, so all of the motion vectors are relevant. Because motion is the best indicator of activity, our technique uses the sum of all motion vectors and the total number of I blocks (a large value typically corresponds to large motions).

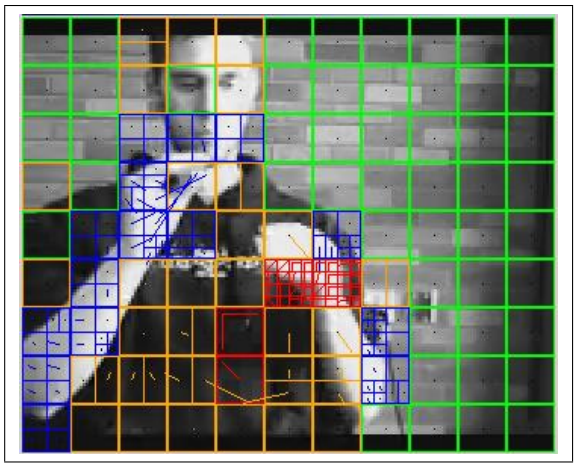
The size and location of hands and face can also give a rough indication of the activity. We detect skin via a simple and well-known RGB-based algorithm [19] that works for many different skin tones. We apply a smoothing filter and ignore "skin blocks" smaller than a certain threshold, as these are usually just noise (in Figure 3(b), we ignore the small blocks above and to the left of the main boxes). We then use the center of gravity, area, and bounding box of the three largest connected components as extra features. These correspond roughly to the face, the right hand, and the left hand, though often the skin-detection is noisy. When the user is not signing, or signing close to her torso, there is often no component corresponding to the left or right hands. In this case we send negative numbers, since that information is useful.

Lastly, the sum of pixel differences between frames is often used as a baseline for motion detection. Figure 3(c) shows the result of subtracting the current frame from the previous one. We test against the sum of pixel differences in our results and incorporate it as a feature.

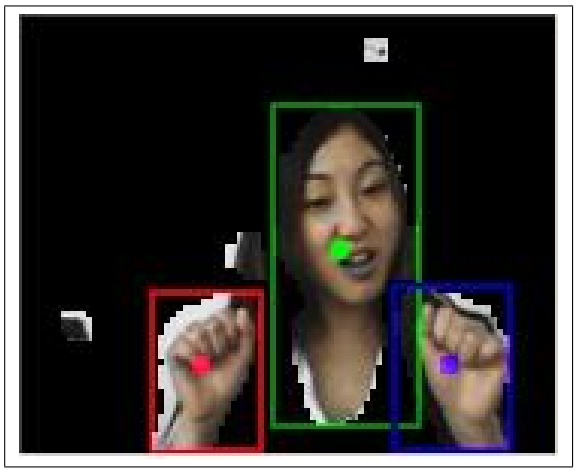
4.2. Machine learning

We use a support vector machine (SVM) [10] , a well-known machine learning technique, to train and test on our features. SVM is an algorithm that, given labeled training data in the form of features and their classes, determines the optimal separating hyperplane that maximizes the distance between the two classes. The hyperplane is not necessarily in the same dimension as the feature space; in fact, it is usually transformed nonlinearly to a higher dimensional space in which greater separation may be achieved.

We use libsvm [6], a publicly available software package, to train and test our data. The kernel function we apply is the standard radial basis function. We improved the results for the SVM by also considering the classification of frames immediately previous to the one to be classified. We looked at the classification returned by the SVM for the three frames before this one, plus the current classification, and returned the majority vote. This sliding window mod-



(a) Macroblock visualization; the lines emanating from the centers of the squares are motion vectors.



(b) The bounding box and centroid visualization.

(c) Difference image. The sum of pixel differences is often used as a baseline.

Figure 3. Feature visualization

eled the temporal nature of the video. We experimented with different weightings on each frame, but found weighting them equally worked best.

4.3. Joint information

The conversational aspect of our videos allows us to incorporate additional information into our training and test sets. Namely, we are able to take features from both streams to aid in our classification. Suppose that two participants, Alice and Bob, are signing to each other over mobile phones. To classify Alice's next frame, we use the feature data from her previous frame plus the feature data from Bob's previous frame. Alice's features make up the first part of the vector and Bob's make up the second part, and we use Alice's label for training purposes. To classify Bob's next frame, we use the same data, except that Bob's features are in the first part of the vector and Alice's are in the second part, and we use Bob's label for training purposes.

5. Results

We compare the results of the SVM on single stream and joint stream data to a very simple baseline differencing technique. A rough measure of motion is the sum of the absolute differences between the current frame and the previous frame. We determine the optimal threshold above which we will classify the frame as signing by training. We can combine data from both streams by simply subtracting the difference of the other stream from the difference of the current stream. Intuitively, this works because if the current user is signing and the other user is not signing, the joint difference will be large, but if the current user is not signing and the other user is signing, the joint difference will be small (negative, in fact). If both users are signing or not signing, the joint difference will be higher if the current user is signing more vigorously.

We extracted features and trained and tested on eight conversational videos, from four different conversations. We divided each video into four parts, trained on three out of the four, and tested on the fourth. We report the overall accuracy on the entire video using this leave-one-out testing method.

Table 1 shows that the single stream methods were all outperformed by the joint stream methods. Furthermore, our SVM technique, using the motion vector information as features, outperformed the baseline method.

6. Conclusion and Future Work

In this work, we quantified the power savings on a mobile video phone of lowering the frame rate during the less important segments of a sign language conversation. By dropping the frame rate to 1 fps at appropriate times, we saved up to 35% of battery life. We then described our technique for realizing those savings by automatically recognizing when the user is signing. We utilized features available "for free" from the encoder, as well as the joint information from both sides of the conversation. We modeled the temporal nature of the video by employing a sliding window. Our method substantially outperforms a baseline method that doesn't use the joint information.

The next step is to perform the recognition within the encoder on the cell phone, over the GPRS network. Currently, MobileASL has been ported to work on several different types of cell phones, and users can carry on video conversations over the WiFi network. We plan to add the frame dropping technique to MobileASL and conduct user studies to determine the intelligibility and irritation level of the conversations when the recognition makes mistakes. In particular, it would be interesting to know if users change their style of conversation when the algorithm mistakenly lowers the frame rate; if they made large gestures to "turn it back on", this could be detrimental to the overall power savings, since both users would be labeled as signing.

We also plan to further our work in activity analysis of sign language video. We would like to recognize finger spelling frames, which might require a higher frame rate. We plan to incorporate a hidden Markov model on top of the SVM, that takes the classification from the SVM as input and outputs its own classification. This would be another way to model the temporal aspects of sign language conversation.

7. Acknowledgments

We gratefully acknowledge the help of Jaehong Chan and Anna Cavender on this project. Thanks also to the National Science Foundation for funding this work.

References

[1] acbPocketSoft. acbTaskMan Version 1.3, April 2007. http://www.acbpocketsoft.com.

[2] L. Aimar, L. Merritt, E. Petit, M. Chen, J. Clay, M. Rullgrd, C. Heine, and A. Izvorski. x264 - a free h264/AVC encoder. http://www.videolan.org/x264.html, 2005.

[3] M. Assan and K. Grobel. Video-based sign language recognition using hidden markov models. In *Proceedings of Gesture Workshop*, pages 97–109, 1997.

[4] B. Bauer and K.-F. Kraiss. Video-based sign recognition using self-organizing subunits. In *Proceedings of International Conference on Pattern Recognition*, volume 2, pages 434–437, 2002.

[5] A. Cavender, R. E. Ladner, and E. A. Riskin. MobileASL: Intelligibility of sign language video as constrained by mobile phone technology. In *Assets '06: Proceedings of the 8th international ACM SIGACCESS conference on Computers*

Video		Single baseline	Joint baseline	Single SVM	Joint SVM
Noisy	Signer 1	36.3 %	85.5 %	**94.5** %	93.3 %
	Signer 2	65.1 %	85.9 %	94.7 %	**94.9** %
Quiet	Signer 1	76.9 %	82.6 %	91.6 %	**92.9** %
	Signer 2	52.3 %	83.2 %	90.3 %	**90.9** %
Noisy	Signer 3	63.2 %	85.7 %	**84.6** %	84.4 %
	Signer 4	77.8 %	89.1 %	90.5 %	**91.7** %
Quiet	Signer 3	49.8 %	80.4 %	86.3 %	**90.8** %
	Signer 4	46.0 %	85.4 %	**93.7** %	92.5 %
Weighted Average		58.9 %	84.6 %	90.7%	**91.4**%

Table 1. Recognition results for baseline versus SVM. The best for each row is in bold. The average is weighted over the length of video.

and accessibility, pages 71–78, New York, NY, USA, 2006. ACM Press.

[6] C.-C. Chang and C.-J. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[7] F.-S. Chen, C.-M. Fu, and C.-L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image Vision Computing*, 21(8):745–758, Aug. 2003.

[8] N. Cherniavsky, A. C. Cavender, R. E. Ladner, and E. A. Riskin. Variable frame rate for low power mobile sign language communication. In *Assets '07: Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, New York, NY, USA, 2007. ACM Press. To appear.

[9] F. Ciaramello and S. Hemami. 'Can you see me now?' An objective metric for predicting intelligibility of compressed American Sign Language video. In *Human Vision and Electronic Imaging 2007*, January 2007.

[10] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[11] Y. Cui and J. J. Weng. Appearance-based hand sign recognition from intensity image sequences. *Computer Vision and Image Understanding*, 78(2):157–176, May 2000.

[12] GSMA. General packet radio service. http://www.gsmworld.com/technology/gprs/class.shtml, 2006.

[13] C. L. Huang and W. Y. Huang. Sign language recognition using model-based tracking and a 3D hopfield neural-network. *Machine Vision and Applications*, 10(5-6):292–307, Apr. 1998.

[14] C. L. Huang and S. H. Jeng. A model-based hand gesture recognition system. *Machine Vision and Applications*, 12(5):243–258, 2001.

[15] K. Imagawa, H. Matsuo, R. Taniguchi, D. Arita, S. Lu, and S. Igi. Recognition of local features for camera-based sign language recognition system. In *International Conference on Pattern Recognition*, pages Vol IV: 849–853, 2000.

[16] International Telecommunication Union. Trends in Telecommunication Reform 2007: The Road to NGN, Sept 2007.

[17] T. Kobayashi and S. Haruyama. Partly-hidden markov model and its application to gesture recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 3081–3084, 1997.

[18] S. C. Ong and S. Ranganath. Automatic sign language analysis: A survey and the future beyond lexical meaning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6), June 2005.

[19] S. L. Phung, A. Bouzerdoum, and D. Chai. Skin segmentation using color pixel classification: Analysis and comparison. *IEEE Trans. Pattern Anal. Mach. Intell*, 27(1):148–154, 2005.

[20] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: A systematic survey. *IEEE Transactions on Image Processing*, 14(3):294–307, 2005.

[21] T. Starner, J. Weaver, and A. Pentland. Real-time American Sign Language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[22] S. Tamura and S. Kawasaki. Recognition of sign language motion images. *Pattern Recognition*, 21(4):343–353, 1988.

[23] N. Tanibata, N. Shimada, and Y. Shirai. Extraction of hand features for recognition of sign language words. In *Proceedings of International Conference on Vision Interface*, pages 391–398, 2002.

[24] L. Wang, W. Hu, and T. Tan. Recent developments in human motion analysis. *Pattern Recognition*, 36(3):585–601, 2003.

[25] M. H. Yang, N. Ahuja, and M. Tabb. Extraction of 2D motion trajectories and its application to hand gesture recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(8):1061–1074, Aug. 2002.