JOURNAL OF
EMERGING INVESTIGATORS

Article

Building a video classifier to improve the accuracy of depth-aware frame interpolation

Surya Jasper[1], Carol Tu[2], Jedrzej Kozerawski[3]
[1] Saint Francis High School, Mountain View, California
[2] Torrey Pines High School, San Diego, California
[3] University of California, Santa Barbara, Santa Barbara, California

SUMMARY

With the current growth in video technology, 4K resolution and sixty frames per second are becoming the industry standards for live, pre-recorded, and animated footage. Although many old movies have a lower frame rate because of insufficient technology, the larger issue lies in the processing of videos. As videos are streamed online, the frame rate can be significantly impaired. So, if we can computationally increase the frame rate, we can dramatically reduce the amount of data being sent to the user. With the recent rising popularity, efficiency, and effectiveness of artificial intelligence, deep learning is undoubtedly the most plausible solution to this problem. Multiple projects have used neural networks to interpolate videos and improve their frame rate. However, since different categories of videos, such as two-dimensional and three-dimensional, can have drastically different color schemes and motion paths, training a single model to handle all of them leads to overfitting, which can be seen in current interpolation algorithms that are specialized to interpolate certain categories of videos. To combat this issue, we researched whether it would be possible to find a single method to perform frame interpolation invariant to the type of video inputted. In this process, we found several pre-existing models that performed well with either 2D or 3D footage, but not both; therefore, we hypothesized that building a video classifier to categorize the input video's dimensionality would thus improve their accuracy. After integrating the classifier with two depth-aware frame interpolation models, we improved the average accuracy to 97.2%.

INTRODUCTION

As the complexities of computer science continue to rise, expectations for the standard of technology grow with it. Video quality is no exception, but it is very time consuming and expensive for creators to produce a high frame rate for their audience. Frame interpolation is a type of video processing algorithm that attempts to increase the smoothness and fluidity of videos by generating frames between existing ones. This was previously achieved through optical flow estimation (1), but more precise techniques are now available following the uprise in computer vision. Frame interpolation has garnered the attention of the deep learning community because of its applications, including frame rate upconversion, slow motion generation, and frame recovery. Some of the methods employed to achieve video interpolation include adaptive separable convolution (2), deep voxel flow (3), and bidirectional predictive network (4). Even with the development of convolutional neural networks on video frame interpolation, creating high-quality images is still demanding due to large instances of motion and occlusions. Furthermore, the methods previously mentioned are only applicable to specific types of video, which is severely limiting. For example, optical flow estimation relies on calculating the difference of velocity between the camera and objects in the scene, so it does not work well in two-dimensional (2D) animations because of the lack of depth and perspective. The more generalized approach we will delve into further is depth-aware video interpolation.

Depth-aware video interpolation is a method that collects the geometrical data of a scene by tracking points in a video and mapping them in three-dimensional (3D) space, as shown in Figure 1 with the depth maps (5). The optical flows represent the object's motion paths in relation to the point of view of the camera. All depth-aware video interpolation algorithms, albeit through different processes, use the geometrical data and motion paths to construct an interpolated frame by averaging the positions of objects between every two consecutive frames. The two implementations of this method that we will focus on in our experiments are the Depth-Aware video frame INterpolation (DAIN) model and the Super SloMo model.

While the DAIN and the Super SloMo models have very different structures, they follow the same general interpolation process. For every two consecutive frames, the models generate depth maps that assign each pixel a numerical value that represents how close the objects are to the camera. They then use the depth data to isolate the objects in the scene and calculate the difference between the positions and rotations of these objects in the two frames. At this point, the

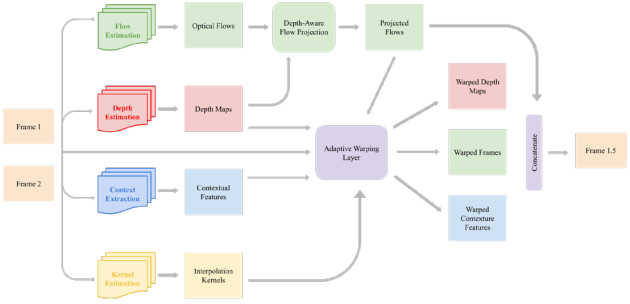


Figure 1: DAIN model video interpolation overview (5). This figure shows the process through which the DAIN model generates an interpolated frame based on depth and color information from two inputted frames. The model repeats this process for every two consecutive frames in an inputted video, producing an interpolated result.

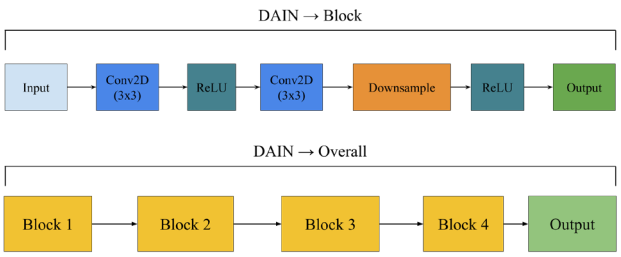


Figure 2: DAIN model diagram. This diagram shows a visual representation of the neural network component of the DAIN model.

DAIN model finds the average of the positions of the objects in the two consecutive frames and generates a new depth map called the interpolated flow (as shown in Figure 2) that contains the predicted depth data in the interpolated frame. However, the Super SloMo model follows a slightly different process to improve accuracy and avoid artifacts (6). Instead of assuming linear motion in moving objects like the DAIN model does, the Super SloMo model uses auxiliary frames (frames besides the two consecutive frames, as shown in Figure 3) in order to better analyze the path and acceleration of the moving objects. The model, then, produces an image to map the change in position. It then uses this equation to predict the new position of these objects in the interpolated frame, rather than simply calculating the average. While this tends to compromise with a longer runtime, it results in a much higher accuracy with 3D clips than the DAIN model. After generating an interpolated flow, both models use the color data from the two consecutive frames to convert the interpolated flow into a colored interpolated frame.

Because of the key differences between the two models, they have a large range of accuracies for different types of video, so neither model is a perfect, versatile solution. The Super SloMo model achieved high accuracies for 3D animated and real time videos with its extensive optimizations, ability to analyze more types of motion, and use of auxiliary frames for reference; however, they result in unnecessary long runtimes and artifacts for 2D frame interpolation. The DAIN model, on the other hand, was created to use depth-aware interpolation for both 2D and 3D videos, but by trying to find a balance between the two while still maintaining high accuracies, their finished model actually interpolated 2D clips with much higher

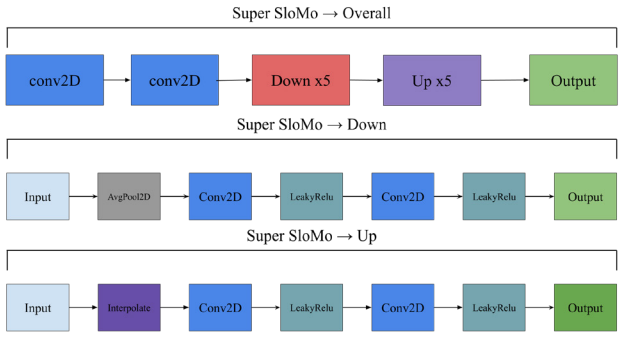


Figure 3: Super SloMo model diagram. This diagram shows a visual representation of the neural network component of the Super SloMo model, which is important in conjunction with Figure 2 since it shows the different approaches the Super SloMo and DAIN model use to interpolate videos, which leads to these varying accuracies.

$$RGBDifference(x,y) = |Ro_{x,y} - Ri_{x,y}| + |Go_{x,y} - Gi_{x,y}| + |Bo_{x,y} - Bi_{x,y}|$$

EQUATION 1. Formula to calculate the difference in red, green, and blue (RGB) values between 2 pixels.

$$isPixelCorrect(f,x,y) = 1 \ if \ (RGBDifference(x,y)) \leq 4, \quad 0 \ otherwise$$

EQUATION 2. Formula to calculate the accuracy of a single interpolated pixel as either 0 or 1.

$$acc = \frac{\sum_{f=1}^{frames} \sum_{x=1}^{width} \sum_{y=1}^{height} \{ isPixelCorrect(f,x,y) \}}{frames \times width \times height} * 100\%$$

EQUATION 3. Formula to calculate accuracy of interpolated video.

$$MSE = \frac{\sum_{x=1}^{width} \sum_{y=1}^{height} (RGBDifference(x,y))^2}{width * height}$$

EQUATION 4. Mean squared error formula for images.

Equations 1-4: *Ro*, *Go*, *Bo*: original Red, Green, & Blue values; Ri, Gi, Bi: interpolated Red, Green, & Blue values; frames: number of frames; width: width of the frame in pixels; height: height of the frame in pixels

accuracies than with 3D clips.

Since both of these models are capable of interpolating videos of different dimensions with varying accuracies, they would not be ideal for use as a standard automatic interpolation tool across all platforms. In light of this issue, we sought to find out if creating a video classifier would improve the overall accuracy of existing interpolation algorithms. Since developing a classifier would utilize the strengths of the DAIN model (3D video) and the Super SloMo model (2D video) while minimizing their weaknesses, we hypothesized that a classifier would improve the overall accuracy of the two individual models. To test this hypothesis, first, we ran tests on the individual accuracies of the DAIN model and the Super SloMo model to determine their strengths and weaknesses. Then, we developed a video classifier and set it up to feed videos to either the Super SloMo model or the DAIN model depending on the result. Lastly, we conducted tests on the final model which consisted of our classifier and the two interpolation algorithms to determine the validity of our hypothesis. The findings supported our hypothesis: the classifier we built was able to interpolate 2D and 3D videos at similarly high accuracy levels.

RESULTS

Both the DAIN and Super SloMo models that we implemented in our project were pre-trained and reported high accuracy for the specific tasks they were tailored to handle.The DAIN model includes 32 layers with a LeakyReLU activation function that serves the purpose of determining which features play a key role in determining the output. We achieved a high accuracy of 98.6% for the 2D animations with this model, but it was 24.3% less accurate when dealing with 3D footage. The Super SloMo model, on the other hand, contains only 12 layers and a ReLU activation function. However, unlike the DAIN model, it seemed to be capable of interpolating either live action or animated 3D footage with close to 99% accuracy, but had trouble interpolating 2D animations, with an inadequate accuracy of 50%.

From the results gathered above, we realized a combination of the strengths from each of the models with a classifier was the most effective way to determine whether the input video was 2D or 3D.

We also collected data on our bilateral filtering algorithm to determine a feasible algorithm for our classifier. After plotting the mean squared accuracies of our filter using the formula in **Equation 4**, we realized that drawing a simple threshold line would work well to classify inputted videos. The

	Real-life videos	3D animated videos	2D animated videos	Average
DAIN	87.5%	74.3%	98.6%	86.8%
Super SloMo	99.3%	98.6%	48.3%	82.1%
Our Classifier	93.8%	84.5%	97.5%	91.9%
Our Final Model	98.5%	96.9%	96.2%	97.2%

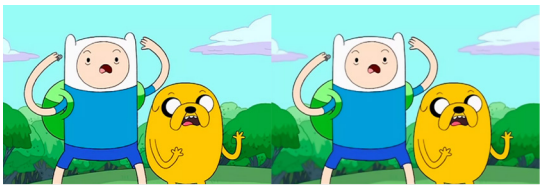
Table 1: Average accuracy based on model used and type of video interpolated (including final model).

results of this experiment are shown in Figure 6. After building our classifier upon this approach, we tested it on the same 1000-video clip dataset, containing half 2D animations and half 3D video clips, that we used to test the DAIN and Super SloMo models. As shown in Table 1, our classifier was able to correctly identify 92% of the videos.

After implementing our finished classifier into the final model, which included the DAIN model and the Super SloMo model, the average accuracy of DAIN increased to 86.8% and 97.2% for Super SloMo (**Table 1**). As shown in Table 1, we tested the accuracy of each individual component of our final model (the DAIN model, the Super SloMo model, and our video classifier) on several different types of videos: clips from 3D and 2D animations and clips from real-life camera footage. The "Classifier" row shows the average accuracy of our classifier alone on frames of videos of the listed types, and the "Final Model" row shows the average video interpolation accuracy of our final model, incorporating our classifier and the two depth-aware models. We calculated these accuracies using the metric provided in **Equation 3**.

Before After

Figure 4: Bilateral filter on real-life image. This figure shows an example of our classifier on a real-life image where the effect of the filter is easily discernible and would lead to a high mean squared error.



Before After

Figure 5: Bilateral filter on cartoon image. This figure shows an example of our classifier on a 2D cartoon image where the difference between the input and output is not as noticeable. Since the before and after images are very similar, the mean squared error would be low, and our classifier would classify it as 2D.

DISCUSSION

Even given that the DAIN model is depth-aware, the creators trained the model on a dataset that consisted partly of 2D animated footage, providing an explanation for its high accuracy with 2D animation. Even though the dataset the creators used is not publicly available, we inferred that it might have contained too many 2D animations, most likely resulting in overfitting and consequently leading to lower accuracies on the 3D clips.

Meanwhile, the SuperSloMo model achieved almost half the accuracy of the DAIN model, which was logical because the Super SloMo model was designed with the intent of being used with 3D video clips, therefore it was trained primarily on 3D footage. This was further verified after reviewing the interpolated results of the 2D animations because we observed sections of the video that were distorted, enlarged, shrunk, and awkwardly colorized. It was obvious that the model was trying to detect 3D objects and depth-revealing color data in a 2D animation, where neither of those features existed.

Although most commercial cameras are capable of filming at 60 frames per second or higher, there are many barriers preventing this from becoming the standard, for which this work proposes a working solution. By combining the strengths of two powerful depth-aware frame interpolation models, we were able to create an improved model capable of interpolating 2D and 3D video types with similar accuracies and with a much higher average accuracy than the two individual models. Manually selecting which interpolation algorithm to use for every video was not feasible, considering how over a billion hours of videos are watched daily on YouTube alone, according to their published statistics. However, the implementation of our algorithm is able to improve the overall quality of videos for viewers and creators alike.

Furthermore, our classifier categorizes images as 2D or 3D a large majority of the time, but it does not have the accuracy required to be a proficient solution. For example, if we fed in 1,000,000 videos into our classifier, around 80,000 of those videos would be incorrectly classified, which would have severely impacted the video quality if we did not use two depth aware methods. With a rise of ongoing study in

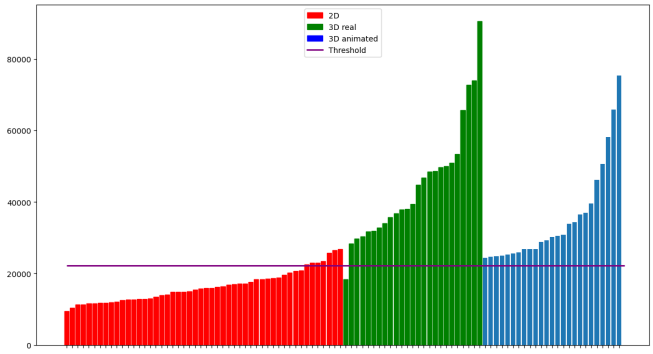


Figure 6: MSE plot for bilateral filter on 100 images with threshold line. This figure shows the mean squared error between the inputted image and the output of our bilateral filtering algorithm for 2D animated images, 3D animated images, and real-life images. The threshold line shows the value our model uses to classify these images, and as the figure shows, this approach works for a good majority of them.

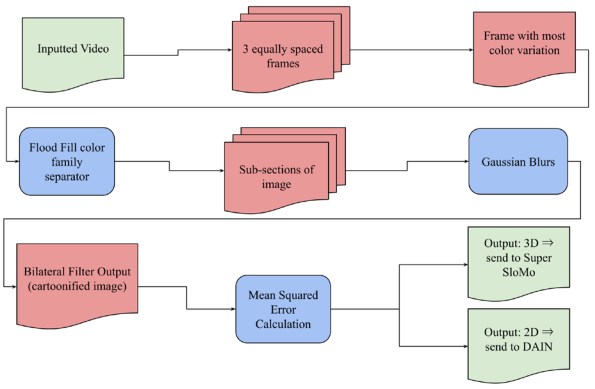


Figure 7. Model diagram of our classifier. This figure is a visual representation of our classifier, including a visualization of our bilateral filtering algorithm.

the field of 2D/3D image classification, we hope to research other methods in order to fine-tune our own bilateral filtering method to increase our accuracy, allowing us to implement an interpolation algorithm built for 2D animation. While our classifier is not yet exact enough to meet the standard for licensing or mass distribution, we could deploy our technology as a platform for users who seek video interpolation for their personal use. Regardless of our current limitations, the practical use of our classifier is valuable for anyone looking to improve the quality of videos and could be a major asset for production companies in the future.

MATERIALS AND METHODS

To set up our workflow, we utilized the available pre-trained DAIN and Super SloMo models and set them up to interpolate a given set of videos to collect data on accuracy and runtime. To test the accuracies, we converted sixty 10-second videos (twenty 2D animations, 3D animations, and 3D real-time videos each) ranging from 60 frames per second (fps) to 15 fps, so we had inputs for the models and control groups to compare the final results of interpolation. After running both models on each of these videos, we then wrote a program to go through every frame of the interpolated result and the original 60 fps video, and then compared the RGB values of every pixel in the interpolated frame with its corresponding pixel in the frame from the original 60 fps video to assess the accuracy of each interpolation.

We initially planned to count how many pixels had the exact same RGB values, but realized that the human eye would not be capable of noticing very minor differences in RGB values. After conducting further research, we realized that out of the 16,777,216 different RGB values, humans are incapable of seeing 6/16 of the possible colors, due to the differences in the individual RGB gradations and human color perception that lead to error. Furthermore, the average human eye can only distinguish the remaining 10,000,000 values only when they are placed right next to each other (7). However, the compared RGB values in our tests were not adjacent, meaning it didn't apply to our color evaluation, and the range of error could be more than the standard 10/16. Thus, we devised an algorithm that considered RGB values as correct so long as the sum of the differences between the red, green, and blue color values for each pair of pixels was

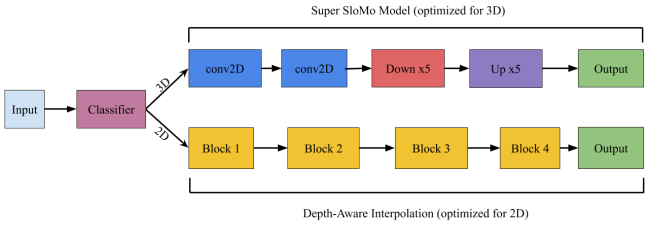


Figure 8. Our model diagram with the Classifier, DAIN model, and Super SloMo model. This figure provides a visualization of our entire model, outlining the process from an inputted video to an interpolated result.

no more than four digits. We decided on this threshold after running multiple trials because it resulted in a high-enough variance for the accuracies of the model, and therefore we were able to categorize our models. RGB values lower than four resulted in accuracies close to 0%, while values over four resulted in almost 100% accuracies, so four was the reasonable middle ground that still provided us with ample RGB values. **Equation 3** expresses a mathematical model for this algorithm.

After running this test, we were able to demonstrate the superiority of the DAIN model versus the Super SloMo model for 2D animations, while the Super SloMo model resulted in higher accuracies than the DAIN model for 3D videos.

In order to fulfill our objective of creating a single system that can interpolate all types of videos for multiple purposes, we would need an efficient video classifier capable of classifying videos as either 2D or 3D without compromising on runtime. Since 2D videos tend to have well-defined outlines and no noise (random pixel variation in brightness and color that are usually seen in real time footage), we decided to extract a frame from each video and apply a bilateral filtering algorithm that increases contrast at edges and blurs sections with high noise. Our classifier could then compare the doctored image with the original and classify the video based on the difference. We determined that this was an appropriate threshold for the following experiments.

We relied on bilateral filtering, a technique that adjusts the intensity of each pixel according to the properties of its nearby pixels using Gaussian blurs (8), to create an animated filter, as seen in Figure 4. These blurs reduce noise and randomness in images, which visibly alter real-life images and 3D cartoons because they lack solid fills but contain noise and gradients. Since 2D animation usually consists of well-defined outlines and solid fills, we conjectured that bilateral filtering would alter 2D images much less than it would for 3D cartoons and live action frames. Figures 4 and 5 show the difference between the effect of bilateral filtering on cartoons and its effect on 3D images. While Figure 4, a real-life image, shows a dramatic difference between the input image and bilateral filtering output, Figure 5, a frame from a 2D animation, has minimal visible difference between the input and output.

However, bilateral filtering usually works best on grayscale images since Gaussian blurs manipulate the color values of surrounding pixels to conform to the pixel being altered. Therefore, for color images with a large variety of hues, the effectiveness of the algorithm would decrease greatly. To counter this issue, we decided to divide the image into smaller regions, each one containing only one color family. To maintain

the fast runtimes, we used the flood fill algorithm. This method is usually used to recursively fill a single-colored section of an image with another color. In our case, we used the algorithm to recursively scan the image and isolate families of colors to prep regions of the photo for bilateral filtering. Afterwards, we would apply Gaussian blurs on each of these sections separately, resulting in a smooth and simplified image.

Based on the difference between the original and filtered image, which was calculated using the mean squared error formula (**Equation 4**), we were able to classify the images as either 2D or 3D.

After conducting an experiment with 100 random images (half of which were frames from 2D animations and the other half from 3D videos) and plotting our results (the mean squared errors), we realized that the 3D images consistently had a considerably higher mean squared error than the 2D images (Figure 6). Upon further analysis, we determined the mean squared error of 22,147 was the optimal threshold to distinguish between 3D images and 2D images, as most of the values above this number were 3D and the numbers below were 2D. With this threshold, we were able to successfully classify videos as either 2D or 3D based on their mean squared error values.

Since the model deals with videos rather than images and we did not want to compromise runtime by running the classification algorithm on each individual frame, we decided to take three random frames from each video and run the algorithm on the frames with the largest diversity of color families through our aforementioned altered flood fill algorithm. This ensured that we would not end up with an image that was neither 2D nor 3D, such as a frame consisting of only one color. We would then feed the video to either the 2D or 3D interpolator depending on the result of our classification algorithm.

Our completed model involves an input layer that is sent through the classifier, which then sends it to either the Super SloMo model or the DAIN model depending on the result, as mentioned in the Materials and Methods section. The video is lastly processed through the proper network (Figure 8) and produces an output with a higher frame rate.

To test the accuracy of our finished model, we used the same formula (**Equations 1-3**) to confirm that the addition of the classifier significantly improved the accuracy.

ACKNOWLEDGEMENTS

Special thanks to Aiwen Xu for her teaching, guidance, and support throughout our period of research. Also thanks to Weilin Sun for aiding in our initial research.

We would like to thank the creators of the Depth-Aware INterpolation model, Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang for providing a two-dimensional interpolation model for our project. We would also like to extend our gratitude to Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz for creating the Super SloMo model, and Avinash Paliwal and Nima Khademi Kalantar for the Pytorch implementation of Super SloMo.

Received: November 30, 2020
Accepted: March 16, 2021
Published: May 24, 2021

REFERENCES

1. Zitnick, Charles, *et al*. "Consistent Segmentation for Optical Flow Estimation." *IEEE Xplore*, 2005, ieeexplore.ieee.org/abstract/document/1544871.
2. Niklaus, Simon, *et al*. "Video Frame Interpolation via Adaptive Separable Convolution." *ArXiv.org*, 5 Aug. 2017, arxiv.org/abs/1708.01692.
3. Liu, Ziwei, et al. "Video Frame Synthesis Using Deep Voxel Flow." *ArXiv.org*, 5 Aug. 2017, arxiv.org/abs/1702.02463.
4. Chen, Xiongtao, *et al*. "Long-Term Video Interpolation with Bidirectional Predictive Network." *ArXiv.org*, 13 June 2017, arxiv.org/abs/1706.03947.
5. Bao, Wenbo, *et al*. "Depth-Aware Video Frame Interpolation." *ArXiv.org*, 1 Apr. 2019, arxiv.org/abs/1904.00830.
6. Paliwal, Avinash, and Nima Khademi Kalantari. "Deep Slow Motion Video Reconstruction with Hybrid Imaging System." *ArXiv.org*, 21 Apr. 2020, arxiv.org/abs/2002.12106.
7. Tabora, Vince. "Human Vision and Digital Color Perception." *Medium, High-Definition Pro*, 11 Feb. 2019, medium.com/hd-pro/human-vision-and-digital-color-perception-91db3b19cc7f.
8. Tomasi, Carlo, and Roberto Manduchi. "Bilateral Filtering for Gray and Color Images." Bilateral Filtering for Gray and Color Images - IEEE Conference Publication, *IEEE*, 1998, ieeexplore.ieee.org/document/710815.