Reprinted from SPIE Vol. 1007—Mobile Robots III
© 1989 by the Society of Photo-Optical Instrumentation Engineers, Box 10, Bellingham, WA 98227-0010 USA

Video Compression for Remote Vehicle Driving

Martin Herman, Karen Chaconas, Marilyn Nashman, Tsai-Hong Hong

Sensory Intelligence Group
Robot Systems Division
National Institute of Standards and Technology
(formerly National Bureau of Standards)
Gaithersburg, MD 20899

ABSTRACT

In order to effectively drive a remote ground vehicle using video images, the operator must be provided with a natural, real-time video sequence and the imagery must be accurate and detailed enough so that the operator can make mobility and survivability decisions. Unfortunately, high data rate communication channels are often not feasible for this task. To accomplish remote driving using a low data rate channel, video compression techniques must be incorporated. This paper discusses the remote vehicle driving problem and describes several video compression algorithms that have been implemented on PIPE, a real-time pipelined image processing machine. The paper then discusses how these algorithms are evaluated on real-world remote driving tests. Finally, advanced techniques for video compression are proposed.

1. Introduction

Remotely driving a ground vehicle involves an operator who sits at a remote control center and views video images that are transmitted from one or more cameras mounted on the vehicle. While observing these images, the operator drives the vehicle by means of driving controls such as a steering wheel, brake pedal, accelerator pedal, etc. These controls generate appropriate driving actuator signals which are transmitted to the vehicle. In order for the operator to effectively drive the vehicle, the video images must be of sufficient quality and be updated as frequently as possible. Full rate video transmission from the vehicle to the operator requires about 60 megabits/sec for 512 x 512 images with 8 bits/pixel at 30 frames/sec. However there are several problems with using the wide communication bandwidth required for such transmission. First, wide bandwidth radio communication requires direct line of sight between the transmitter and receiver. This is not feasible in realistic outdoor scenarios where vehicles are likely to be driven behind hills and mountains and therefore hidden from direct view by the operator station. Wide bandwidth links are also relatively expensive. Further, even if such a link were available, full rate video would use up a large part of the bandwidth allocations. This would be particularly true if there were many vehicles being operated simultaneously, where full rate video might fill up the entire communication spectrum. Fiber optic links, which have wide bandwidth communications capabilities, also have several problems, including limited ruggedness, difficulties in deployment and retrieval, and the problem of repairs.

Many of these difficulties can be overcome by utilizing narrow band radio links which have communication bandwidths on the order of 100 kilobits/second. Since the full video required for teleoperation cannot be provided using the narrow band links, efficient and effective techniques of real-time video compression offering compression ratios of 500:1 to 1000:1 must be developed. Compression ratios this large require more than conventional reduction techniques. They require an examination of the fundamental requirements for remote driving.

2. Requirements for Remote Driving

A vision system that is used for remote driving can be evaluated in terms of how well it permits the remote operator to perform vehicle mobility operations. There are various factors that affect vehicle mobility. The first involves the ability of the operator to make trafficability and movement decisions. These include:

1. Local obstacle detection. The vision system should allow the operator to detect the following classes of local obstacles:

This paper was prepared by United States Government employees as part of their official duties and is therefore a work of the US Government not subject to copyright. Identification of commercial equipment in this paper is only for adequate description of our work. It does not imply recommendation by the National Institute of Standards and Technology, nor that this equipment was necessarily the best available for the purpose.

a. Depressions such as craters, holes, trenches and ruts.

b. Solid objects on the ground such as rocks, logs, and debris.

c. Vegetation such as heavy brush.

d. Tall objects such as trees and telephone poles.

2. Local surface classification. The vision system should allow the operator to determine the kinds of surfaces surrounding the vehicle. These surfaces would include swampy or soft soil, concrete, sand, mud, and grass. Also, this includes snow, ice, or water on the surface. Finally, these would include ponds, lakes, and rivers.

3. Local surface orientation determination. The vision system should allow the slope of the local terrain to be determined visually, although this information can also be provided by a level detector or inertial navigation system on the vehicle.

4. Local landmark recognition. The vision system should allow the operator to recognize local landmarks such as trees, rocks, telephone poles, etc. The operator needs this information to perform local path planning.

5. Local path planning. The vision system should provide enough information for the operator to perform local path planning. The information required includes the relative location of local obstacles, the location of classes of local ground surfaces and their orientations, and the location of local landmarks. Further, the operator requires the spatial relations among the obstacles, landmarks, and the ground surfaces. For example, spaces between objects must be determined, as well as relative positions between objects not all of which are simultaneously in the field of view of the cameras.

6. Local path following. The vision system should provide enough information so as to be able to follow terrain features such as roads, rivers, or vegetation.

A second factor that affects vehicle mobility involves the ability of the operator to visually maintain a sense of the global vehicle location relative to the background and landmarks. These include the ability to identify landmarks and the background, as well as the ability to track landmarks over time. Notice that the ability to maintain global orientation can also be provided to the operator with the aid of an inertial navigation system or a compass on board the vehicle. This allows the vehicle position and direction to be displayed on a map in front of the operator.

A third factor that affects vehicle mobility involves the ability of the operator to visually maintain a sense of the motion parameters of the vehicle, such as speed, acceleration, and turning rate. Again, this can also be provided by means of other sensors on board the vehicle, such as a speedometer or accelerometer.

Finally, the vision system should be able to provide the above capabilities under all weather and daylight conditions, and under a variety of vehicle speeds.

In order to satisfy the requirements set forth above, two additional requirements should be met. First, the video images should be displayed in real time. This means that the operator should have the appropriate visual information quickly enough so that he can drive the vehicle interactively. Driving is typically a servoing operation which requires a fast loop involving transmitting driving commands to the vehicle, getting fast visual feedback, and using this feedback to transmit additional driving commands.

A second requirement is that the video sequence that appears on the operator's monitor should seem natural (e.g., smooth and continuous). This would minimize the training requirement for the operator, as well as limit adverse side effects to the operator (such as headaches and nausea).

The requirements set forth above should be satisfied whether the vehicle is driven with full video or with compressed video. A disadvantage of using video compression algorithms is that they take processing time, and many algorithms result in degraded imagery. A challenge for video compression techniques is therefore to meet all the requirements set forth above, that is, to provide real-time video, to provide a natural video sequence, and to provide imagery which is accurate and detailed enough so that the operator can make mobility decisions.

3. Approach

Our approach to the problem of video compression for remote driving is to use a hybrid method which combines image processing techniques (i.e., techniques whose input is an image and whose output is a compressed image), transform techniques (such as the discrete cosine transform), and temporal frame rate reduction (i.e., transmitting fewer than 30 images per second). The sequence of events as they would occur in the system is as follows. Images are obtained from one or more cameras mounted on-board the vehicle. These images then undergo compression using the hybrid technique. After the compressed code is transmitted over a communication link to the operator station, it is decompressed so as to result in a sequence of full resolution images. The compressed images will be transmitted over the communication link at a rate of at most a few per second. However, we want the images to be displayed on the operator's monitor at

30Hz. This requires an extrapolation procedure, which will be provided by a real-time image warping processor. This procedure involves generating a realistic simulation of the imagery that would appear for the given camera (to be discussed further below).

Our plan for this project has been to first implement video compression algorithms on the PIPE real-time image processing machine (to be described in more detail below). PIPE is excellent for quickly developing, testing and modifying algorithms. PIPE was then integrated with a remote control vehicle system and the algorithms were evaluated on real-world remote driving tests. Initial results of these tests are described below. After a subset of algorithms have been chosen based on such tests, they will be reimplemented on special-purpose image processing boards which will reside on-board the remote vehicle.

In the remainder of this paper, we describe the PIPE machine, we discuss compression algorithms implemented thus far on PIPE, and we provide experimental results of real-world tests. A companion paper [2] presents the topic of video compression in more detail and also provides details on the PIPE implementations of the compression algorithms.

4. PIPE

PIPE (Pipelined Image Processing Engine) is a multi-stage, multi-pipelined image processing device that was designed for real-time robot vision applications. PIPE was conceived and designed at the National Institute of Standards and Technology (formerly National Bureau of Standards) [3] working jointly with Aspex, Incorporated. It is currently commercially available through Aspex. PIPE will accept images from a video camera at field rates -- 60 times per second. PIPE's basic cycle rate is 1/60 second.

The PIPE system is composed of up to eight identical modular processing stages, each of which contains two image buffers, five look-up tables, three arithmetic logic units, and two neighborhood operators. Images are transferred from stage to stage at field rate (60 images per second) by three concurrent pathways. The forward path allows traditional pipelined and sequential processing. The recursive path from a stage output back to its input allows feedback and relaxation processing. The backward path from one stage to the previous stage allows for temporal operations. It also allows a hypothesis image to be compared with an input image. The images in the three paths can be combined in arbitrary ways on each cycle of a PIPE program, and the chosen configuration can change on different cycles. In addition, six video buses allow images to be sent from any stage to any one or more stages.

Images can be processed in any combination of four ways on PIPE: point processing, spatial neighborhood processing, sequence processing or Boolean processing. Point processing can be either a function of one or two input images and includes simple arithmetic and logical operations such as scaling, thresholding, converting number systems, etc. More complex arithmetic operations, trigonometric operations, comparisons, rotations, etc. can also be performed.

PIPE can perform up to two 3 x 3 neighborhood convolutions on each stage in parallel. Both neighborhood operators operate on the same image input, but can perform different neighborhood operations. The neighborhood operators can be either arithmetic or Boolean.

Sequential processing works on a set of multiple images, e.g., sequences of images over time, a stereo pair of right and left images, or multi-resolution images.

When performing Boolean processing, each pixel of information is considered to be composed of eight independent bit planes, which are operated upon simultaneously. The neighborhood operators can be applied in a Boolean mode, where the output is the combination of the 3 x 3 neighborhood using local operations on each of the eight bit planes.

5. Compression Techniques Implemented on PIPE

A number of data compression algorithms have been developed and demonstrated on PIPE. These include grey level quantization, non-maxima suppression, foveal-peripheral simulation, image differencing, histogram slicing, binning, Laplacian pyramids, Poisson interpolation, and linear predictive coding. The following briefly describes each of these methods. We characterize the effectiveness of these algorithms by their entropy value. Details on how entropy is calculated are described in [2]. Note for now that the lower the entropy value, the greater the compression. Further details on the implementations on PIPE may also be found in [2].

Grey scale quantization involves reducing the resolution of each pixel in the image. As represented on PIPE, an image pixel contains 8 bits. However, image resolution and contrast remain acceptable when three or even four low-order bits are dropped. Thus the number of bits required to transmit an image can be reduced by 37.5% or 50% respectively. When implemented on PIPE, the update rate of this method is one cycle or 1/60th of a second. The entropy measurement is dependent on the number of low-order bits that are dropped. It ranges between a value of 6.74 for a full resolution image to 3.10 for an image in which four bits have been dropped.

Non-maxima suppression is an image processing method which results in a binary edge image in which all edges are one pixel wide. On PIPE, a Sobel edge operator is first applied to the input image. Then all edge points that are not locally maximal in the edge gradient direction are eliminated. The update rate for this algorithm is two cycles (1/30th of a second), and the entropy measure is 0.37. The compression ratio is very high since only one bit of information per pixel is required.

The foveal-peripheral simulation is based on the biology of human vision. In humans, there is a very small area of acute vision, surrounded by areas of degraded vision. Two methods were written based on this idea. In the first (Figure 1a), a square window of the image is displayed with full 8 bit resolution. This square is surrounded by concentric window bands containing 6 bits, 4 bits, 2 bits, and finally 1 bit of resolution. The full resolution window can be repositioned and resized to meet the user's requirements. The implementation of this method on PIPE has an update rate of two cycles, or 1/30th of a second. For a square window measuring 60 x 60 pixels, the entropy measure is 2.02.

The second foveal-peripheral algorithm utilizes the concept of multi-resolution image processing. In multi-resolution processing, a full sized image is successively sampled and reduced in resolution by a factor of two. Thus a 256 x 256 image is reduced to a 128 x 128 image which is reduced to a 64 x 64 image, etc. Using this technique, an arbitrarily sized and positioned square window is displayed at its full resolution (Figure 1b). It is surrounded by a window band obtained from its next level of resolution which in turn is surrounded by a band at the next level in the pyramid. Each level of the pyramid is successively more blurred, but requires fewer bytes of information. The implementation of this method on PIPE has an update rate of four cycles (1/15th of a second). For a square window measuring 60 x 60 pixels, the entropy measure is 4.05.

Image differencing is an effective compression technique when there is relatively little motion between successive scenes in a sequence of images. For a stabilized camera mounted on a vehicle, the distant background will appear not to move between successive images, while the foreground will appear to move. The difference image is generated by subtracting an image at time t_{n-1} from the image at time t_n . All stationary regions of the image are eliminated and only areas of motion are visible. Thus much less information than the full image sequence need be transmitted. The full image sequence can be reconstructed from the difference images. In practice, good results have been achieved by transmitting a full image every 8 seconds, and difference images every 1/30th of a second. At reconstruction, the difference images are summed with the originally transmitted image to form a sequence of full images as follows:

$$t_0+(t_1-t_0)+(t_2-t_1)+ \cdots +(t_n-t_{n-1})=t_n .$$

The update rate on PIPE for this method is one cycle (1/60th of a second) and a typical entropy measure for a difference image is 2.01.

Image compression and reconstruction using decimation and Poisson interpolation [4] is achieved on PIPE by passing the input image through a Laplacian neighborhood operator and thresholding the resultant image such that strong edges retain their grey level value while homogeneous regions are mapped to zero. This decimated image is transmitted to the remote operator station. Reconstruction involves multiple iterations of interpolation of the decimated image in regions which had been homogeneous. The update rate for the decimation is two cycles, and the update rate for the reconstruction is N cycles where N represents the number of iterations performed. Good results have been achieved by using six iterations.

Linear predictive coding is the process by which the grey scale values at each pixel in two sequential images are used to predict the grey scale value at a future point in time. Theoretically, this prediction is based on the assumption that the intensity values at any given pixel location vary only slightly over time, when the length of time is relatively short. Since the time between sequential images is small, the extrapolation of the grey scale values using a linear function is assumed to provide an accurate prediction. The delay time for the algorithm on PIPE is 1/10 second (6 cycles).

Binning is a term used to describe a general class of methods also known as histogram transformation. Once the histogram of an image is obtained, ranges of grey scale values can be grouped together to a single value based on a number of different criteria. Three methods of performing this grouping were investigated. One method was to construct each bin from an equal range of grey scale values. Another method was to have each bin contain an equal integral portion of the histogram curve. Histogram slicing, the third method, is accomplished by dividing the histogram of an image according to the most populous grey scale values.

The Laplacian pyramid method of encoding an image [1] is based on both predictive and transform methods of image compression. The prediction of a pixel's value is based on a local Gaussian-weighted average of surrounding pixels. These predicted values are subtracted from the original values so that a measure of the predicted error is produced. The result is that only the predicted error, which requires fewer bits to be represented, and the low-pass filtered image, which can be sampled, need to be encoded. This concept is repeated to produce a pyramid-like data structure. After the encoded pyramid structure has been transmitted, it may be used to reconstruct the original image. On PIPE, the entire encoding and decoding of a single image requires 51 cycles (0.85 seconds). The total entropy of all transmitted images is

6. Experimental Results

This section describes the performance of several video compression algorithms running on PIPE during real-world remote driving experiments. The remotely controlled vehicle system we used was developed by AAI Corporation. The vehicle itself was a six-wheeled off-road recreational vehicle. A color camera mounted on the vehicle obtained 512 x 512 pixel images with 8 bits each of red, green, and blue. The camera had a wide field of view -- 90 degrees. The full video signal was transmitted to the remote operator through a radio link, and displayed on a monitor in front of the operator. The operator could remotely control the steering, acceleration, and brakes through the radio telemetry link. The operator did not have pan and tilt control of the camera. The maximum vehicle speed was about 30 miles per hour.

Our experiments were performed on a very gray, overcast day. PIPE was set up at the operator station, using as input one of the three color channels returning from the vehicle. PIPE then performed real-time video compression on the input images, and the results were displayed on the monitor and used by the operator to drive the vehicle. The tests were performed with an experienced driver, who gave an impression of his ability to drive using the output of several PIPE compression algorithms. All algorithms were tested on the AAI grounds in Hunt Valley, Maryland, both in a paved loading dock area surrounded by buildings and on an uneven grassy field. We should note that the operator had considerable experience driving this particular vehicle on the given test terrain. The algorithms tested and the impressions from the driver are enumerated next.

1. When PIPE was connected to the operator station, the original 512 x 512 x 24 color video was switched to a 256 x 256 x 8 monochrome video. The visual contrast on the monitor dropped tremendously, but our test driver could still drive the vehicle without serious problems.
2. Grey scale quantization involves reducing the resolution of each pixel in the image. When one low-order bit was dropped, the driver's response was a bit slower than with 8 bits per pixel. When two low-order bits were dropped, there was not much additional change. When three low-order bits were dropped, there were stepped contrast gradations, and the driver claimed to have better depth perception. The dropping of four low-order bits resulted in much slower response -- the stepped contrast was worse and the driver felt that everything was out of focus, that objects seemed to be flowing into one another, and that object boundaries were not sharp enough.
3. The non-maxima suppression algorithm results in a binary image containing thinned edges. The driver felt that this was unusable for driving. In the loading dock area, only edges of buildings were visible.
4. We tried both of the foveal-peripheral algorithms. One involves a square window displayed with full 8 bit resolution surrounded by concentric window bands containing 6, 4, 2, and 1 bit of resolution, respectively (Figure 1a). The driver felt that this was not really usable, partly because it was necessary to have a portion of the front of the vehicle in the field of view, and the algorithm did not provide this with enough resolution. The driver stated that if the bottom center portion of the screen, containing the front of the vehicle, had been used as the central high resolution window by the algorithm, it might have been usable for driving.
5. The second foveal-peripheral algorithm, involving lower spatial resolution concentric bands (Figure 1b), obtained very similar results. The driver mainly used the high resolution area of the screen, and completely ignored the rest of the image. It was most useful when the high resolution area covered the front of the vehicle.
6. The image differencing algorithm, with a full image being transmitted every 30 images, resulted in a much slower response by the operator than full video.
7. The Laplacian pyramid algorithm was not usable by the driver for two main reasons. First, the 850 millisecond update rate of the algorithm on PIPE was too slow to be effective for driving at reasonable speeds. Second, the output image continually changed even when the vehicle was stationary, thus further confusing the driver. This was probably caused by noise in the images.

The primary difference between driving on the paved loading dock area and on the open field was that features in the loading dock area were easier to recognize. On the open field, obstacles and features of the landscape were much more difficult to recognize. The main difficulties that were encountered during the experiments, particularly when driving in open terrain, were (1) global vehicle location relative to the background and landmarks was very difficult to determine, (2) the slope of the local terrain was very difficult to determine, (3) ditches, gullies, rocks, and other obstacles were difficult to distinguish, (4) range from the vehicle to objects and terrain features were difficult to determine. Object and terrain feature segmentation was very difficult in monochrome images. This was particularly true in the open field where the main way of discriminating between objects and the field was by color -- the field was covered with green grass and weeds, while objects and obstacles were brown, silver, or red. We believe that the wide field-of-view camera may have contributed towards some of the difficulties. The wide view results in lower resolution of features in the image, as well as reduced parallax. The latter leads to poorer depth perception. Some of the difficulties may also have resulted from the gray, overcast sky, since the poor lighting may have contributed to low contrast in the images.

It seemed quite important to be able to see a portion of the front of the vehicle in the image. This was useful partly in establishing global vehicle location relative to the background, but mainly in perceiving clearance for the vehicle and perceiving depth of objects.

One of the conclusions that we drew from these experiments is that remote driving using full color video is difficult as is. Almost any compression technique which provides a high enough compression ratio will result in degraded imagery. This will make remote driving even more difficult.

These experiments constitute only an initial set of tests. We plan to perform additional tests with all the techniques described above. These tests will involve different kinds of cameras, mounted in different positions on the vehicle, and under different lighting conditions.

7. Plans for Effective Video Compression

Based on our initial experiments, we also plan to examine a combination of the following elements to attain video compression for remote driving:

1. rather simple image processing compression techniques, such as spatial and temporal sampling, dropping low-order bits, or image differencing,
2. transform techniques, such as the discrete cosine transform,
3. realistic simulation of imagery at the operator station, allowing images to be transmitted at rates much slower than video rates,
4. graphic cues overlayed on the imagery at the operator's monitor, allowing information such as the front of the vehicle, the horizon line, or parallel lines extending forward from the sides of the vehicle to be graphically displayed.

The first two elements above were discussed earlier. The third element is related to the work of Noyes and Sheridan [6], who introduced the technique of using a locally situated forward-in-time telerobot simulator for remote robot operation. This technique was developed to handle a time delay between the operator and the telerobot. With this technique, control signals are transmitted simultaneously to the simulator system and to the remote telerobot. The simulated video is then superimposed onto the return video of the remote teleoperation. This allows an operator to move the controls and immediately see the effects of the control commands without having to wait for the return video. We feel that this technique can be very powerful in the context of video compression for remote driving. The time delay between the operator and the remote vehicle is due to two factors, the processing time of the compression algorithms and the time delay due to temporal sampling of the video sequence -- an image is transmitted to the operator only a few times per second, or even once every few seconds.

The most difficult problem in pursuing the simulation approach is generating a realistic simulation of the video sequence as viewed from the cameras on board the vehicle. One approach to this is a technique that we will call *non-adaptive image warping*. This technique, mentioned in a previous section, involves transmitting an image (called the *master* image) relatively infrequently (perhaps once every few seconds), but then warping the master image in real time to generate a sequence of simulated images for the operator [5]. These images would approximate those that would appear in the camera. When a new image arrives from the vehicle, it becomes the master image and the procedure is repeated. Because the warping technique is non-adaptive, all warping parameters are predetermined, resulting at times in very poor simulated images.

Another approach to the simulation is one that we will call *adaptive image warping*. Figure 2 shows the concept behind this approach. As with non-adaptive image warping, there is a master image transmitted periodically, and a sequence of images generated by warping the master image. In the figure, let c_0 be the pose of the camera when the master image is taken, and let c_i be the pose of the camera at some future point at which we want to generate a simulated image. For some pixel p_0 in the c_0 image, we know that the point P_s in the scene corresponding to this pixel lies along a ray from the camera focal point through the pixel p_0 out into the scene. In fact, the point P_s is at the intersection of this ray with the first surface it meets. If we know the position of this surface from a scene model, then we can determine P_s. The pixel p_i in c_i corresponding to pixel p_0 in c_0 is simply obtained by the intersection of the line containing P_s and the focal point of c_i with the image plane of c_i. Pixel p_i will then be given the same grey level value as pixel p_0. In practice, the point p_0 corresponding to pixel p_i will lie in between pixels. Therefore, the grey value assigned to pixel p_i will be obtained by interpolating pixel values in a small neighborhood around p_0. When this procedure is carried out for every pixel in c_i, we have the resulting warped image.

Briefly, the operation of adaptive image warping is as follows. A video image is transmitted from the remote vehicle to the operator station at the rate of, say, 1 frame per 3 seconds. Time-tagged vehicle navigation data is transmitted from the vehicle continuously at 30 Hz. These data are used to estimate the position and orientation of the camera in the scene during the simulated views. Time-tagged 3-D scene information is transmitted once every 3 seconds. Simulated images are then generated at 30 Hz from the master image and from the navigation data and scene information. These images are displayed to the operator and provide the approximate current vehicle scene. Because the vehicle's pose is always known to a high accuracy, any vehicle motion can be reflected instantly in the operator display. A Kalman filter is applied to the navigation data so that a smooth extrapolation is obtained to the point in time at which the image is to be displayed at 30 Hz.

Of fundamental importance to this approach is the ability to acquire, represent, and efficiently access a scene model. The scene model should contain both large scale terrain features (hills, valleys, plateaus, etc.) as well as small features (roads, trees, rocks, ditches, etc.). Such a model can be derived from a combination of sources such as digital elevation terrain data, maps of natural features, maps of man-made features, three-dimensional information extracted from camera images, and rangefinder data. The more accurate the acquired scene model, the better the visual simulation. Therefore, even a simple scene model such as the assumption of a parallel ground plane will result in a simulation that might be useful.

The fourth element that we propose for attaining effective video compression involves graphic overlays which include the following:

1. The horizon line displayed on the screen. This may be used by the operator to visually extract the real horizon line when the video image quality is poor. If the scene model and navigation data (as required for adaptive image warping) are available, then the position, orientation, and shape of the horizon line on the screen can be calculated. If the navigation data are available without a reasonable scene model, then an artificial horizon can be created. An artificial horizon is known to be of great assistance to vehicle piloting in the case of flying, particularly instrument flying. Remote driving has many similarities to instrument flying.
2. Parallel lines displayed on the screen which extend forward from the sides of the vehicle [7]. This may help the operator to determine whether there is clearance to maneuver between obstacles. Again, a good scene model will allow accurate calculation of the position, orientation and shape of these lines on the screen. Without a reasonable scene model, these lines can be approximated under the assumption that the ground is an extended plane.
3. The outline of the front of the vehicle displayed on the screen. A clear appearance of the vehicle on the screen (or painted right below the screen) is necessary for effective remote driving.

8. Summary

This paper has described the remote vehicle driving problem and has presented criteria for evaluating vision systems used for remote driving. Several algorithms were then described that have been implemented as possible candidates for a hybrid video compression system. The algorithms have been implemented on the PIPE real-time image processing machine. Details on these implementations are presented in a companion paper [2]. The PIPE has been integrated with a remote control vehicle system and these algorithms were evaluated by means of real-world remote driving experiments. The results of these experiments were presented. Finally, an advanced video compression approach using adaptive image warping was proposed.

Acknowledgements

This work is supported by the U.S. Army Laboratory Command, Human Engineering Laboratory. Special thanks go to Mr. Charles M. Shoemaker of the Human Engineering Laboratory for his direction and guidance on this project. We also thank Mr. James S. Taylor and Mr. Steven D. Yeung of AAI Corporation for arranging the use of their vehicle for our testing, and Mr. Joseph M. King of AAI Corporation for serving as our test operator.

References

1. Burt, P. J., and Adelson, E.H. "The Laplacian pyramid as a compact image code." *IEEE Transactions on Communications,* vol. COM-31, no. 4, 1983, 532-540.

2. Herman, M., Chaconas, K., Nashman, M., and Hong, T.-H. "Low Data Rate Remote Vehicle Driving." *Proc. Third IEEE International Symposium on Intelligent Control,* Arlington, VA, August 1988.

3. Kent, E.W., Shneier, M.O., and Lumia, R. "PIPE (Pipelined Image Processing Engine)." *J. Parallel and Distributed Computing*, 2, 50-78, 1985.

4. McMillen, R. Personal communication. Hughes Aircraft Co., 1988.

5. Narendra, P. Personal communication. Honeywell Systems and Research Center, 1988.

6. Noyes, M., and Sheridan, T. "A Novel Predictor for Telemanipulation Through a Time-Delay." *Proc. Annual Conference on Manual Control,* NASA Ames Research Center, Moffett Field, CA, 1984.

7. Taylor, J.S. Personal communication. AAI Corporation, 1988.

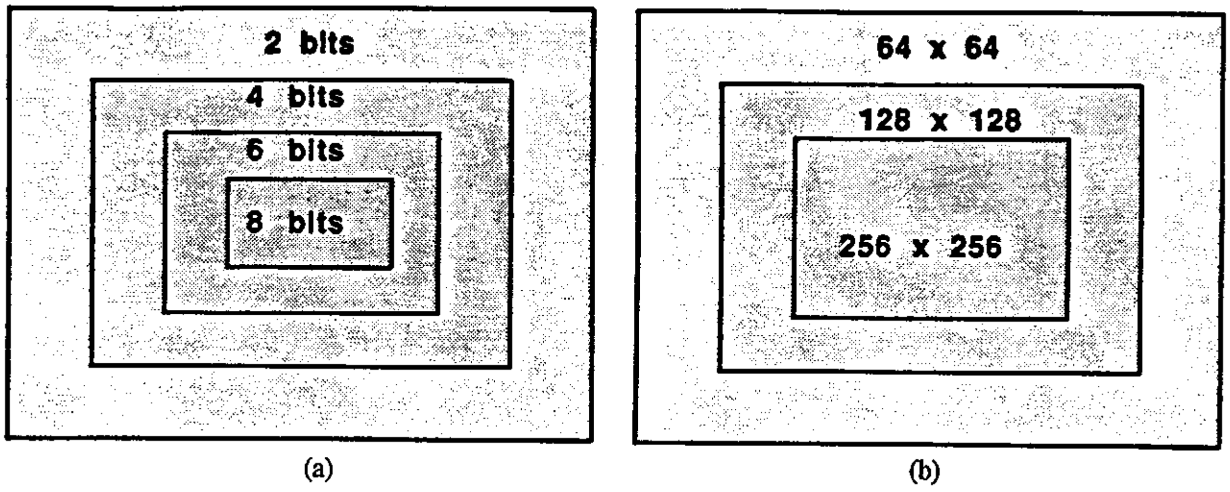

Figure 1. Two compression techniques based on foveal-peripheral vision.

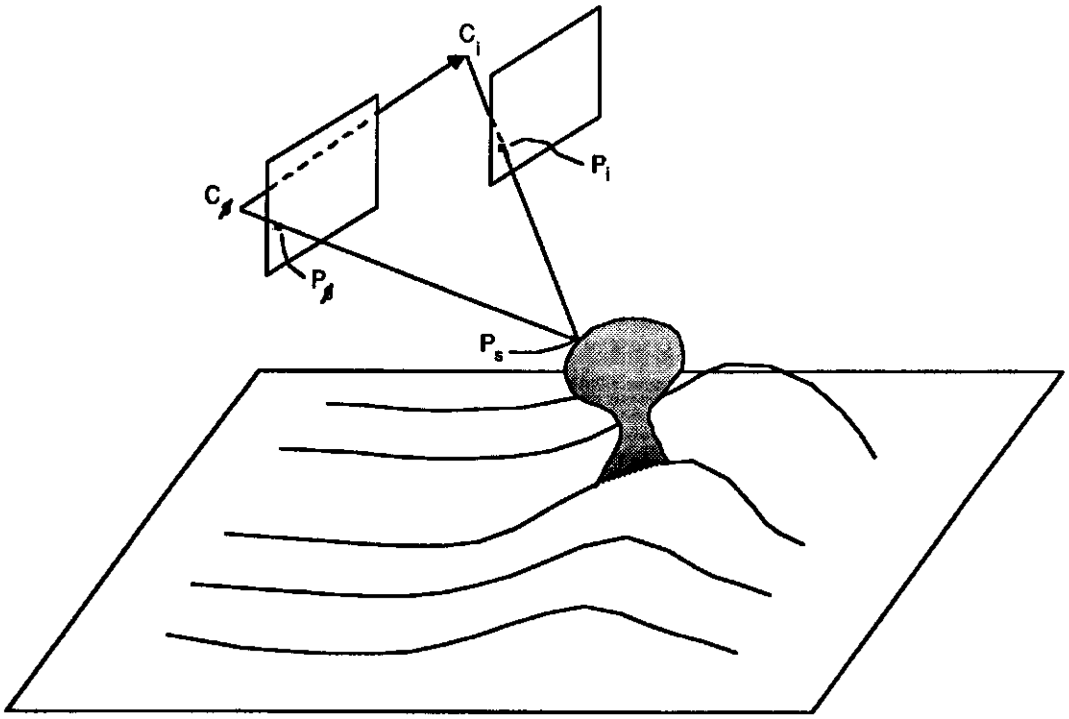

Figure 2. Adaptive image warping.