Scene understanding to aid in limited-bandwidth remote driving

Final Report - CS 231A

Hayk Martirosyan
Brady Jon Quist

March 18 2015

Abstract

We present an effective two-stage algorithm to adaptively compress video streams for limited-bandwidth remote driving. The first stage applies various types of scene understanding to create image rankings that identify areas important for driving, which are combined into a single global ranking matrix. The second stage uses the global ranking matrix to intelligently pre-compress the stream, which is then passed into a H264 codec for final compression.

Our results show a 65% size reduction of an already fully-compressed video stream, where the resulting video almost exclusively loses information about areas not important for driving. This is impressive because the H264 format already uses many complex methods to compress data and is an industry standard (YouTube, Vimeo, etc.). We accomplish this by matching our adaptive pre-compression methods with those used by the codec. In particular, we perform adaptive quantization on 8x8 discrete cosine transform blocks. We believe this is a promising approach to enable remote-driving using available communications infrastructures.

Further work would incorporate additional ranking methods, parallelize the process, and perform testing and validation in real-world scenarios.

1 Introduction

Progress on autonomous driving promises to make our transportation infrastructure more efficient. In particular, huge efficiency gains could be made in the transportation of goods by using vehicles without humans in them. These vehicles would have a fraction of the components of passenger vehicles, and could be designed without crumple zones, seats, steering wheels, infotainment, airbags, door and window controls. Instead, they could consist of a simple electric drive-train, a sensor cluster, and a cargo bay. In this type of logistics infrastructure, the hardware is lighter and cheaper, there are no human drivers, and the system can be intelligently automated and running 24/7, 365.

However, fully autonomous vehicles are not yet ready to fulfill this vision. Existing solutions operate well in pre-mapped regions and good conditions, but break down in adverse weather and in the presence of unexpected events or unknown terrain. One alternative which acts as a gateway to this ultimate goal is to use remotely-driven vehicles. The hardware remains the same, but the vehicle is operated by a human in a driving simulator at a remote location. This approach retains many of the efficiency benefits of the autonomous version while keeping the flexibility and judgement of human drivers.

One of the key challenges with this approach is maintaining a robust communication stream that enables remote driving. In order to reliably operate a remote vehicle, the driver must have a low-latency, high-fidelity, and wide-coverage stream of audio and video. Existing infrastructure has good coverage in many areas, but bandwidth can be very limited. To accommodate this, we propose that many parts of a typical stream have no importance to driving capability. For example, the driver does not care about details of the sky, trees, or objects very far away. A nearby object moving quickly across the view, however, is extremely important.

We propose to rank the components of a vehicle's stereo vision stream in terms of their importance to remote driving capability, with the goal of enabling limited-bandwidth transmission of the stream.

2.1 Review of Previous Work

Previous work by Herman[3] and DePiero[1] explore methods for low-bandwidth remote driving. Herman uses a combination of several methods such as quantization, image differencing, and a foveal-peripheral algorithm, while DiPiero focuses on a Laplacian Pyramid approach. Both develop a full stream processing pipeline that was implemented and tested for remote driving.

While conceptually interesting, these works are from the early nineties and simply lacked the computing power to achieve significant online results. The resolutions on which they operated are nearly impossible to use for urban driving. Herman reports that simple gray-scale quantization was their most effective method. However, their output was a monochrome 256x256 image. DeNieto claims to have achieved remote driving of a Humvee at 15mph on a dirt terrain, but their output video is at 3-6 Hz and has a latency of one second. Many of the approaches in these works are sound, but they simply did not produce outputs that would work for urban remote driving.

2.2 Extensions on Previous Work

We extend previous work by presenting a structured method of adaptive compression that incorporates previous and novel methods of image ranking, and present results on high-quality images and video, using modern image and video codecs.

3.1 Technical Summary

We present a two-stage video processing pipeline that uses adaptive ranking and compression. The first stage takes each input image and outputs a global ranking matrix that specifies how important each pixel is for the purpose of remote driving. The global ranking matrix is a combination of rankings by various computer vision methods that highlight important features, as detailed in the next section. Figure 1 shows features that may be considered when determining the ranking. The second stage uses the ranking matrix to adaptively compress the stream such that we preserve more details about the highly-ranked parts of the image.

Our global ranking matrix was composed of two main sub-rankings, the first based on lane detection and the second based on the disparity map. While we did not implement them, further sub-ranking matrices could be added from image segmentation, colors of interest, moving objects, etc. To combine these sub-ranking matrices into one global ranking matrix, we took the maximum across all sub-ranking matrices. This was based on the idea that if a region was found to be important by any sub-ranking matrix, then that region should experience less compression than regions that were never found to be important.

The second stage allows us to compress different regions by varying degrees based on the global weighting matrix. Different compression methods that we implemented and explored includ bilateral filtering, Gaussian spatial smoothing, color quantization, temporal smoothing, and Discrete Cosine Transform (DCT) compression in the spatial and temporal regimes.

3.2 Technical Details

3.2.1 Ranking Matrix Opportunities

While we lacked the time to implement all of them, there are several methods to be considered for extracting information that can be used to create the global ranking matrix.

Lane detection[1] - Localizing the road and lane markers is extremely important to understanding the scene. The road vanishing point is the single most important point of the image, and naturally segments the image. The triangle formed by the lane markers identifies the immediate terrain that the vehicle is crossing and the driver should be paying attention to.

Stereo disparity map - Calculating a depth map of the scene from the stereo cameras is extremely useful for understanding, because we can draw conclusions about how important various features are by their distance from the vehicle. We can tie this depth data with recognition methods to identify features.

Image segmentation - Segmenting the image into known regions provides an easy way to enable variable compression. These objects might include the sky, road, grass, median, trees, buildings, etc. Based on the object classification, we could determine the compression level and frame rate that different portions of the video need to be transmitted at.

Feature recognition - More advanced feature detection for entities like vehicles, pedestrians, and animals would highlight key areas to pay attention to. This could potentially enable transmission of feature metadata instead of pixel data, to be reconstructed in the form of augmented reality for the remote driver.

Visual flow - Going beyond single-image methods, investigating the temporal differences of the images in the stream can tell us information about the movement of objects. We could say, for example, that objects moving quickly across the screen are extremely important to watch for.

Edge detection - General edge detection can be important for compression, with minimal required knowledge of the scene. In general, regions of the stream without significant edges are much less likely to be important to transmit in high-resolution.

3.2.2 Compression Opportunities

After the global ranking matrix has been calculated, we can apply different smoothing and quantization techniques that together make the images and/or video stream more compressible.

[1] As part of our efforts, we created a tutorial for lane detection using OpenCV 3.0 and Python 3.4. It can be viewed online at `http://nbviewer.ipython.org/github/hmartiro/project-mvq/blob/master/python/notebooks/lane_detection.ipynb`

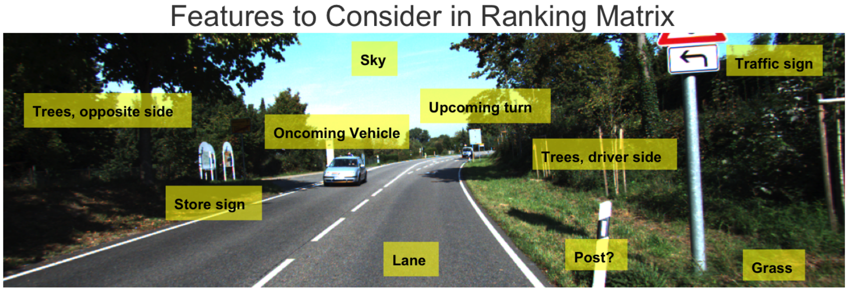

Figure 1: Features that will have varying levels of importance for a remote driver.

While there are many more techniques that could have been used, we have listed here those that we did experiment with.

Bilateral Filtering - In order to compress the images, we are interested in preserving the edges, which hold important information, while reducing the noise. The bilateral filter is a non-linear filter that preserves edges while smoothing out the fine details and noise that are less important for driving.

Gaussian Spatial Smoothing - Certain image and video codecs find it difficult to compress images that have sharp discontinuities. While we want to maintain the edges in regions that are important for driving (such as on the road), regions that are not important can be smoothed to increase the compressibility without affecting the remote driving ability.

Color Quantization - Other image and video codecs, particularly those that are lossless (such as PNG and GIF) can do a much better job compressing images that have a small fixed palette of colors, even if those colors are arbitrary. By choosing a specified number of colors from the images or video that best represent the regions that are important, the video feed may become more compressible. Choosing a color palette for smart color quantization is effectively done by using a k-means or a median cut algorithm.

Temporal Smoothing - Many video codecs exploit the relationships between neighboring image frames to avoid sending redundant data. By temporally smoothing the data, there is a possibility of reducing the bandwidth required to transmit the video stream while maintaining its usability for remote driving.

Discrete Cosine Transform - Other image and video codecs (e.g., JPEG, MPEG, etc.) exploit the DCT to transform the image/video in blocks and then quantize them, thus making them compressible. This enables varying levels of compression based on how aggressive the quantization is. Furthermore, DCT quantization can be performed in both spatial and temporal regimes to further affect the compressibility.

4 Experiments

As this project focuses on the Computer Vision component of this problem, we used real world data that are publicly available and are well calibrated [2]. This alleviated some of the potential difficulties associated with the setup (e.g., alignment, syncing the image frames from two separate cameras, etc.) and allowed us to get further in the computer vision aspects of our proposal. Our software uses Python 3.4 and OpenCV 3.0.

Stage One - Ranking

Our initial effort focused on the detection of lane markers and their vanishing point, which we judged the most important information to know. Our approach is to identify the lane marker lines and calculate the vanishing point as their projected intersection.

We initially tried Canny edge detection followed by a Hough transform for line detection of the lane markers. We found that the results often included spurious lines aligned with the outline of a row of trees or other irrelevant features. To overcome this, we implemented a special lane marker detection filter as described by Nieto [4]. We further adapted it to include a binary erosion algorithm to eliminate noise. Note that we ignored the top 1/3 of the image because it should never be needed to detect the lanes.

We then explored line detection using this special lane marker filter. Our initial experiences showed lines connecting dense regions of scattered points. To combat this, we switched to a probabilistic Hough method as implemented in OpenCV. We defined a minimum line length and maximum line gap that greatly reduced the number of noisy lines. These parameters can be reasonably estimated ahead of time. In order to determine the vanishing point of the road, we implemented our own RANSAC algorithm. Our implementation looked at the intersection of two randomly selected lines and found the number of detected lines which passed within a specified distance to that intersection. We have found this method to be robust to various types of images and that it reliably estimates the vanishing point as seen in Figure 2.

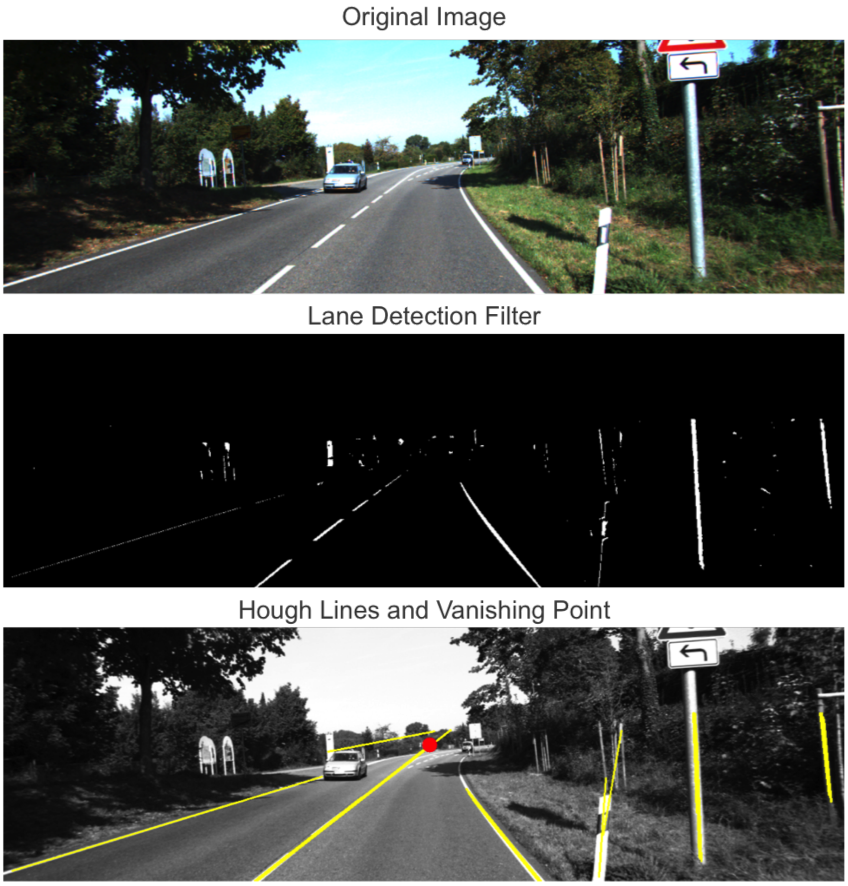

Figure 2: The progression of our lane-detection algorithm.

We also created disparity maps from our data, which give us very useful information about what image components are near and far from the driver. We take stereo images from a color camera pair, and feed them into OpenCV's semi-global block matching algorithm[2]. We tuned several parameters including the window size, disparity count, and correspondence uniqueness ratio to find a good fit for our application. Finally, we use a closing algorithm with a small elliptical kernel on the disparity map to fill in gaps and create a smoother image. The disparity map that we found is identical to the disparity map sub-ranking matrix shown in Figure 3.

Using the found vanishing point, we first created a sub-ranking matrix that included the entire lane up to the vanishing point. This region was given a high ranking because knowing what is in the vehicles path is extremely important for remote driving. Note that while not shown here, we later adjusted the triangle so that the top point was higher in the image. This helped prevent important features like slight lane turns, the tops of cars, and signs from being overly compressed. Another sub-ranking matrix was also created using the disparity map, with the idea that features that are closer to the vehicle should be more important for the driver to see. These two sub-ranking matrices were then combined using the maximum of the two to create a global ranking matrix as seen in Figure 3.

[2]OpenCV StereoSGBM `http://docs.opencv.org/java/org/opencv/calib3d/StereoSGBM.html`

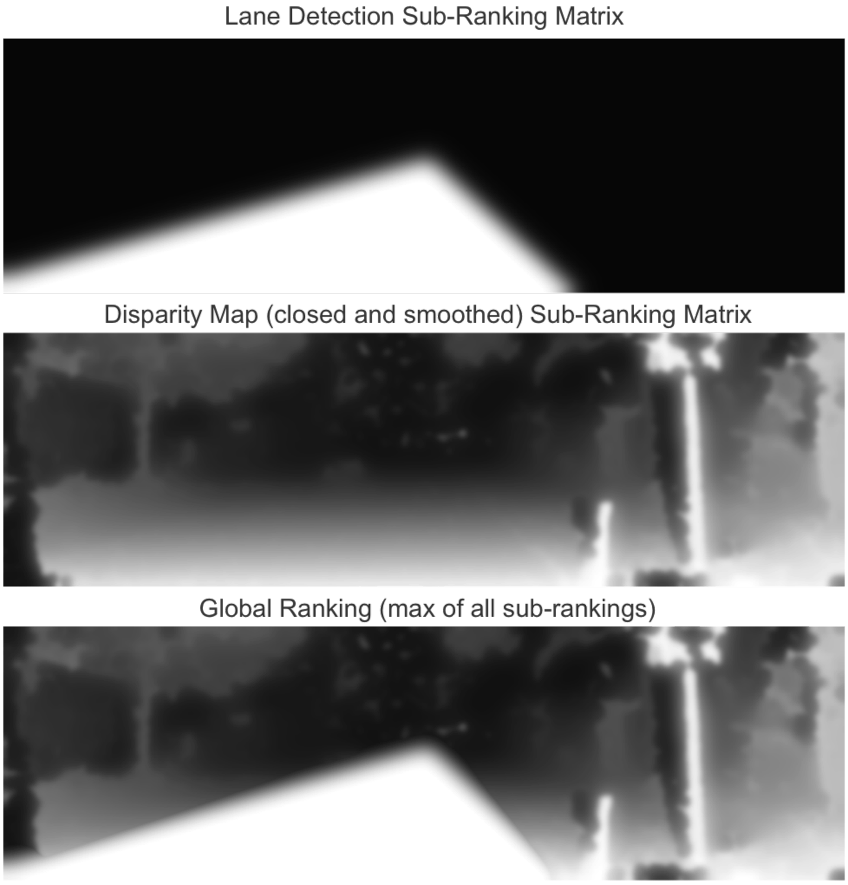

Figure 3: The ranking matrices calculated in stage one.

Stage Two - Adaptive Compression

Once we had obtained a global ranking matrix from stage one, we began focusing our efforts on stage two, which focuses on making the images/video more compressible without sacrificing their usefulness for remote driving. In order to analyze the compressibility more quickly, we first tested the compressibility on individual images and then used that knowledge to work on compressing our video stream.

Knowing that the PNG image format provides lossless compression, we first looked into creating compressed PNGs so that we could determine how much of the file-size reduction was due to our stage two algorithm. As we began the project we were aware that fine details generally make an image less compressible and that smoothing an image, while retaining important edges, should make an image more compressible. To smooth the image, we first performed Gaussian smoothing on the entire image and then created a weighted average of the original and the smoothed versions. The weights were performed on a pixel-by-pixel basis according to the ranking matrix. This preserved all of the fine detail in important regions of the image while smoothing less-important regions. This was followed by filtering the entire image by a bilateral filter that preserved the edges, but removed details that were not helpful for driving or compressibility. The image after both of these stages can be seen in Figure 4.

As the PNG format is lossless, we found that performing color quantization (even for arbitrary colors) makes the image more compressible. This can happen because severe color quantization has the ability to smooth an image (even though it may have sharp discontinuities at color boundaries) and also because the presence of fewer colors enables better compression of the

Figure 4: The top image shows original input image after it has been Gaussian smoothing (where the weight of the smoothing was determined by the global ranking matrix) and then filtered using the edge-preserving bilateral filter. The bottom image shows the top image after color quantization using a color palette of only 15 colors, where the colors were chosen based on the global weighting matrix.

colors themselves using algorithms such as Huffman Coding. We experimented with using both k-means and median cut to find the color palette. In both instances, the input colors were chosen randomly from the image based on the weighting matrix. This ensured that the colors needed to represent the important areas were more likely to be chosen. We further adapted the algorithm by passing in blocks of important colors (such as red, orange, and yellow) so that the images would always be able to accurately display road signs, construction markers, and other important information. In our experiments we found that the median cut algorithm was both faster and slightly more stable, so we chose to use that for further analysis. Using the image computed after Gaussian smoothing and bilateral filtering as an input to the color quantization, we were able to create the bottom image seen in Figure 4 after selecting only 15 colors.

The original input image (see Figure 2), when saved as a PNG, had an original file size of 903 KB and a compressed file size of 297 KB. The bottom image in Figure 4, which utilized our custom ranking-dependent smoothing and quantization had a non-compressed file size of 109 KB and a compressed file size of 48 KB. This represents an impressive compression of 88% and 84% for the non-compressed and compressed images, respectively.

After we verified our initial work, we realized that there are other formats which appear to take better advantage of the spatial information, while also giving flexibility in how lossy the output stream can be. Additionally, we noticed that our output video streams didn't achieve nearly as good of compression levels as the images. This is in part because most common video codecs utilize compression techniques that weren't perfectly aligned with our proposed stage two approach.

One common compression technique used in both image and video codecs is to quantize the Discrete Cosine Transform taken on different channels of the images (commonly the luminance and chrominance channels). We therefore implemented a block 8x8 DCT quantization method, where we utilized scipy's DCT function and the JPEG Standard, Annex K Quantization tables, which give varying levels of quantization depending on a multiplication factor. This multipli-

Figure 5: The top image shows the image after ranking dependent DCT quantization. The bottom image shows an image that goes through, in order, color quantization, gaussian smoothing, bilateral filtering, and then DCT quantization.

cation factor could then be adjusted for each block based on the ranking matrix. The original image after the DCT quantization (using a quality factor of 20 that was multiplied by the ranking matrix) can be seen in Figure 5. Figure 5 also shows an image that first goes through color quantization, then gaussian smoothing, then bilateral filtering, and finally DCT quantization. Note that we also implemented an 8x8x8 DCT quantization method, to additionally quantize temporal changes, that showed marginal improvement on video compression. However, we decided not to use this method because a DCT in the temporal domain inherently introduces undesired lag into our remote driving solution.

After testing the entire approach (i.e., color quantization, gaussian smoothing, bilateral filtering, and DCT quantization) on the video stream we found that the results looked somewhat foggy and the compression level in video, which takes advantage of temporal similarities, wasn't as large as it was for images. Consequently, we decided to stop using the guassian smoothing and color quantization because they were the two methods that most contributed to the fogginess and also weren't directly aligned with the compression schemes of our codec. We did still use the bilateral filter across the entire image because it does a good job removing fine details that aren't important while still retaining the edges. The DCT quantization then became the primary method of compressing different regions by different amounts as defined by the ranking matrix. Because we were able to use a codec that exploited the 8x8 DCT quantization method, this led to large compression factors while still allowing us to retain detail in regions that were important for remote driving.

A comparison of the original image and our final output image, computed using a global bilateral filter and a ranking dependent 8x8 DCT quantization, can be seen in Figure 6. Here we see that the bilateral filter does a great job preserving edges while smoothing fine textures such as that seen on the asphalt. Additionally we see that important features (e.g., the lane markers in the road and the "left turn" sign on the right side the road) are only slightly compressed, while less important features (e.g, the trees that are far away) are quantized much more aggressively. For our video compression comparison we used a video stream which included 370 frames from

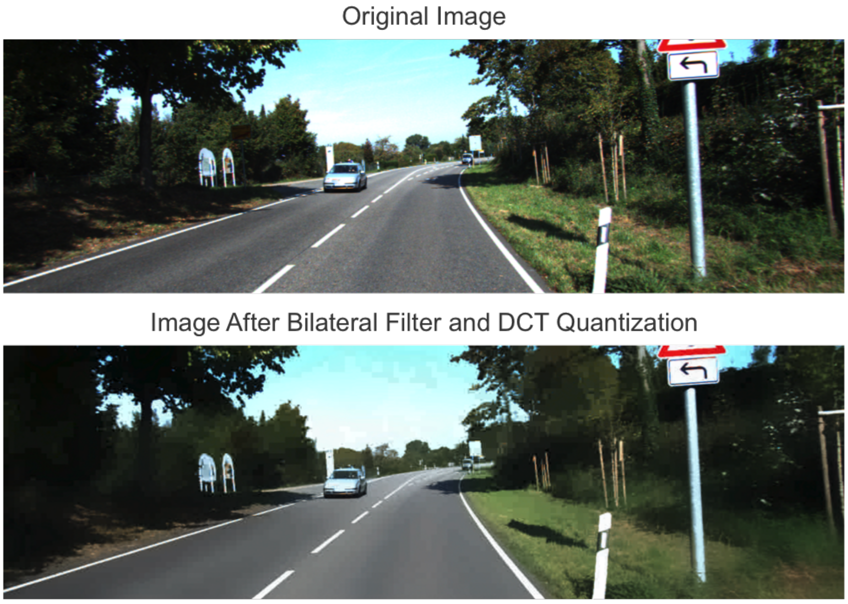

Figure 6: The original image (top) compared to our final output image (below). The output image first passes through a global, edge-preserving bilateral filter and then ranking dependent DCT quantization. Note that all important details (such as the road and sign) are not overly compressed while less important details (such as the trees that are far away and the sky) are much more compressed. The output image is therefore able to be compressed by 65% while still maintaining all information important for remote driving.

the same dataset as the original image and performed the actual compression using the H264 codec (as implemented in ffmpeg) while enforcing the use of 8x8 DCT quantization. The video stream that was first processed with the bilateral filter and then the ranking specific DCT quantization was an impressive 65% smaller than the corresponding original video stream. This is particularly noteworthy because the compressed video stream retains almost all details that would be important for safe operation of the remote vehicle.

Discussion

One of the drawbacks of our current work is that our algorithm could overly compress unnoticed turns or signs that are far away. While these drawbacks are present in our implementation, they are not drawbacks of our overall two stage strategy and can be overcome in various ways. For example unnoticed turns could be identified using standard recognition schemes. Even without identifying unnoticed turns, the turns could be compressed less by modifying our triangle to include a large rectangle in the middle of the image that would cover the regions where most turns generally lie. Signs could also be detected using either standard computer vision recognition methods or by looking for vertical poles in the disparity map. Once detected, the signs could be marked as important in the ranking matrix and would therefore not suffer from aggressive compression.

We note that our implementation was run on laptops without exploiting multiple threads. It is therefore not real-time and as such would currently not work for a remote driving application. However, all of the operations that we perform are highly parallelizable and could likely be implemented on a GPU or FPGA to get real-time processing that could be performed inside the remotely driven vehicle.

5 Conclusion

We present an effective two-stage algorithm to adaptively compress video streams for limited-bandwidth remote driving. The first stage applies various types of scene understanding to create image rankings that identify important areas, which are combined into a single global ranking matrix. The second stage uses the global ranking matrix to intelligently pre-compress the stream, which is then passed into a state-of-the-art codec for final compression. The result is that for a given bit-rate, our method provides a more useful stream for remote driving than with generic compression. Further, we are able to scale the amount of compression to fit within the available bandwidth.

The global ranking matrix we tested takes into account the lane and road vanishing point, as well as a depth map of the view. There are many more possibilities that can be easily plugged-in as a sub-ranking matrix. For adaptive compression, we explored a bilateral filter, Gaussian smoothing, color quantization, and a block discrete cosine transform with a weighted quantization matrix.

A key point is that we do not use custom representations of the data, but adapt our loss methods such that an industry-standard codec does the final compression for us. This is a differentiating factor from previous work. In our video tests, we used the H264 format, which is very effective and currently used by YouTube, Vimeo, and many modern services. We were able to achieve enhanced performance by tailoring our pre-compression methods to use the DCT transform also employed by the codec.

Using this method, we were able to bring down the size of an already fully-compressed H264 video stream by 65%. This is impressive because the H264 format already uses many complex methods to compress data, and yet our resulting video almost exclusively loses information about areas not important for driving.

Further work would incorporate additional ranking methods, such as sign detection, object recognition, texture-based segmentation, and visual flow. The loss step could be further adapted and tuned to take advantage of the more complex methods of video codecs. Tests on many types of driving data should be done to improve robustness. The algorithm should be very parallelizable, and future work should take advantage of this. Finally, experiments should be performed for actual remote driving with outside participants in order to validate the results.

We believe this is a promising approach to enable remote-driving using available communications infrastructures.

References

[1] Frederick W DePiero, Timothy E Noell, and Timothy F Gee. A video transmission system for low-bandwidth remote driving. Technical report, Oak Ridge National Lab., TN (United States), 1993.

[2] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[3] Martin Herman, Karen Chaconas, Marily Nashman, and Tsai-Hong Hong. Video compression for remote vehicle driving. In *1988 Robotics Conferences*, pages 136–145. International Society for Optics and Photonics, 1989.

[4] Marcos Nieto. Lane markings detection and vanishing point detection with opencv. `https://marcosnietoblog.wordpress.com/2011/12/27/lane-markings-detection-and-vanishing-point-detection-with-opencv/`.