Influence of Rain on Vision-Based Algorithms in the Automotive Domain

by

Yazan F. Hamzeh

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in the University of Michigan-Dearborn
2021

Doctoral Committee:

Associate Professor Samir A. Rawashdeh, Chair
Assistant Professor Abdallah Chehade
Assistant Professor Jaerock Kwon
Assistant Professor Alireza Mohammadi

Yazan F. Hamzeh

yhamzeh@umich.edu

ORCID ID: 0000-0002-0795-1429

Dedication

I would like to dedicate this dissertation

to my wife Eman, who was forever supportive of me while pursuing my dream and took over more than her fair share of responsibility, to keep our small family going.

to my daughter Zeina and son Jude, who were anxious about concluding my journey. We made it kids!

to my mother Nabiha, who taught me by example that perseverance was the only sure way to achieve your goal, no matter how hard the circumstances were. Thank you, Dr. Faraj.

Acknowledgments

I would like to thank Dr. Samir Rawashdeh for his excellent guidance and support throughout my research work. I sincerely thank my dissertation committee, In alphabetical order: Dr. Abdallah Chehade, Dr. Jaerock Kwon, and Dr. Alireza Mohammadi, for their valuable time and advice. Special Thanks to Amanda Donovan and Michael Hicks for all the administrative support you provided over the years.

Table of Contents

List of Tables

List of Figures

Abstract

The Automotive domain is a highly regulated domain with stringent requirements that characterize automotive systems' performance and safety. Automotive applications are required to operate under all driving conditions and meet high levels of safety standards. Vision-based systems in the automotive domain are accordingly required to operate at all weather conditions, favorable or adverse. Rain is one of the most common types of adverse weather conditions that reduce quality images used in vision-based algorithms. Rain can be observed in an image in two forms, falling rain streaks or adherent raindrops. Both forms corrupt the input images and degrade the performance of vision-based algorithms. This dissertation describes the work we did to study the effect of rain on the quality images and the target vision systems that use them as the main input. To study falling rain, we developed a framework for simulating failing rain streaks. We also developed a de-raining algorithm that detects and removes rain streaks from the images. We studied the relation between image degradation due to adherent raindrops and the performance of the target vision algorithm and provided quantitive metrics to describe such a relation. We developed an adherent raindrop simulator that generates synthetic rained images, by adding generated raindrops to rain-free images. We used this simulator to generate rained image datasets, which we used to train some vision algorithms and evaluate the feasibility of using transfer-learning to improve DNN-based vision algorithms to improve performance under rainy conditions.

Chapter 1

Introduction

1.1 Motivation

Autonomous driving gained a lot of momentum in the last few years, and many Original Equipment Manufacturers (OEMs) started commercializing some self-driving vehicles, with different levels of autonomous driving capabilities. Ford Motor Company, for example, has completed a 500,000-mile test in April 2021 of its "BlueCruise" hands-free highway driving technology. Other OEMs offer similar hands-free driving assist technology, such as GM's "Super Cruise", and Tesla's "Autopilot". Advanced Driver-Assist Systems (ADAS) are the building blocks for autonomous driving and are already integrated into most new vehicles. For best performance and improved robustness, these ADAS applications usually employ two or more sensing systems, including Radar, Lidar, ultrasonic, and cameras. On-board cameras are very attractive sensing options, due to their relatively low cost and the rich amount and type of information they can provide. Research on the effect of adverse weather conditions on the performance of vision-based algorithms for automotive tasks has had significant interest. Safety and robustness are critical aspects of all automotive applications. This means that these applications must operate under all driving conditions and, in case of failure, need to fail safely, for both vehicle occupants and others sharing the road with the target vehicle. Vision-based systems in the automotive domain must similarly operate at high performance and robustness

status, including operating under all weather conditions. It is generally accepted that adverse weather conditions reduce the quality of captured images and have a detrimental effect on the performance of vision-based algorithms that rely on these images. To guarantee a safe and robust operation of vision-based systems, the effect of adverse weather conditions must be addressed, either by denoising the source of input, or making the system less sensitive to such input noise. Rain is a common and major source of image quality degradation. In our research, we focused on studying the effect of rain in its two forms, falling and adherent, on image quality, and extended this to the overall effect of rain on the operation of vision-based systems that consume these corrupt images. Our research work included the following aspects:

1.2 Falling Rain Simulation

One of the main hurdles to test the performance of vision-based systems in the automotive domain is the lack of reliable and large datasets, of matched rained and clear images. this is due to the uncontrollable and unpredictable nature of rain, and the irreversible degradation it causes to input images. We developed a falling rain simulator, that can generate synthetic rained images, with rain streaks added to the image scenes.

1.3 Falling Rain De-Raining Algorithm

Falling rain reduces visibility by occluding objects in the image scene it may also reduce illumination levels in the vehicle environment which degrades the performance of vision-based systems. We developed an algorithm that detects falling rain in rained images and restores images to the state that is very close to the rain-free images. We used datasets that were generated through out falling rain simulator to test this de-raining algorithm.

1.4 Adherent Raindrop Simulator

We developed an adherent raindrop simulator to generate images with added synthetic raindrops. Adherent raindrops are close to the image plane which means they occupy bigger segments of the not only occlude parts of the scene, but also quality It occludes parts of the input image used in vision-based algorithms and blurs background texture in regions covered by them.

We used datasets that were generated through this simulator for retraining DNN-based algorithms in some later research work we conducted.

1.5 The Correlation Between Image Quality and The Performance of Target Vision Algorithm

While most prior work in the field of rain detection and removal focuses on the image restoration aspects, they typically do not provide quantitative measures of the effect of degraded image quality on the performance of image-based algorithms. Rain introduces uncontrollable and irreversible distortions in input images, that even the state-of-the-art de-raining algorithms cannot fully correct. We studied the effect of degradation to rain on input images, and it relates to the performance of the target vision system that consumes them. Based on that, we presented a quantitative measure of the correlation between image quality and performance of vision-based algorithms.

In the following sections, we will describe these different research areas in detail.

Chapter 2

Background

This chapter provides a background on some image processing techniques and image quality metrics that were used in this rain detection and removal research work.

2.1 Image Processing Techniques

2.1.1 Image Blurring

Image blurring may be a by-product of an image denoising operation, intended to remove high-frequency noise from images. Gradient Decent (GD) of image pixels is calculated based on their intensity levels, and pixels with GD greater than a predefined threshold are considered noise pixels. 2D lowpass filtering has the effect of averaging the intensity level of pixels in a filter window, which reduces the intensity GD, and thus reducing high-frequency noise in an image. consists of pixel areas with intensity gradient decent

Figure 2-1 shows how a 2D Gaussian filter can be used to smooth create a blurring effect in an image.

Figure 2-1: The blurred image (right) is created by applying a blurring effect using a 2D Gaussian filter with a standard deviation of 4.5 to the original image (left). Sharp edges are smoothed out, which can be noticed on the traffic sign text becoming unreadable in the blurred image.

The blurring effect is also used to simulate the effect of camera lenses on captured images. On-board general-purpose cameras used in automotive applications are usually of fixed optics that allow light rays emitted from far objects to be projected on the camera sensor, which represents the image plane. The focal point of light rays emitted from closer objects fall behind the image plane, creating on the image plane what is known as blur circle or disk of confusion. Figure 2-2 shows a representation of the blur circle caused by optics with fixed focal length. The diameter of the blur circle is given by:

$$\epsilon = \frac{\Delta g f^2}{O(g - \Delta g)(g - f)} \tag{2.1}$$

where O is the camera aperture size, f is the focal length, g is the distance between the camera lens and far objects and Δg is the difference in distance between far and near objects. Adding a blurring effect to simulated objects in an image improves visual accuracy and potentially improves the learning performance of deep-learning vision applications.

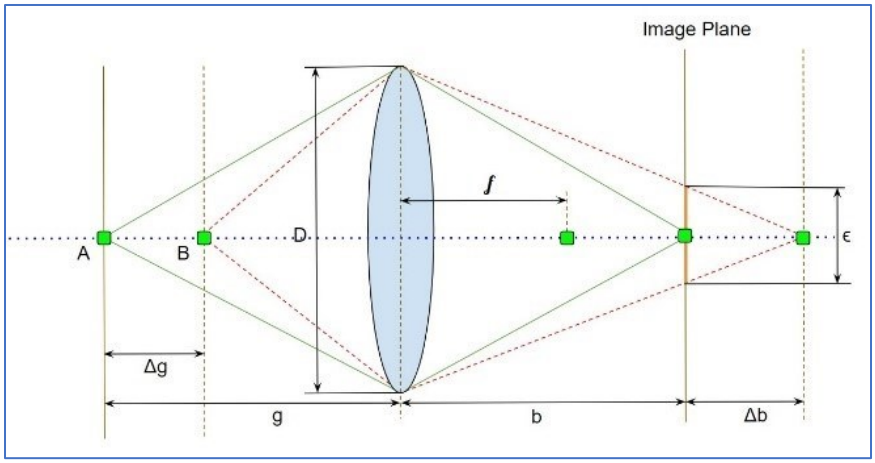


Figure 2-2: On-board cameras are equipped with optics that create focused images of far objects in the environment. Light rays from closer images are projected to the image plane on a disk, commonly known as the blur circle or disk of confusion.

2.1.2 Blob Detection

In image processing, a blob is an acronym for **B**inary **L**arge **OB**jects and is defined as a collection of adjoint connected pixels that share a common attribute (e.g., intensity level, color) that distinguishes them from their neighboring pixels. Secondary attributes are usually extracted from these blobs, that are used to describe the image content in a condensed model, leaving the structural integrity of the image intact. These secondary attributes include, for example, blob size in pixels, the orientation of the smallest ellipse that encompasses the blob area, and the width and breadth of such an ellipse.

As described by Kong et al. [1], blob detection algorithms can be either based on derivative expressions or local extrema in intensity landscape. Laplacian of Gaussian (LoG) [2] was proposed by is the earliest derivative-based blob detection algorithm. An intensity image function, $f(x, y)$ is first convolved with a two-dimensional Gaussian function

$$G(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} \tag{2.2}$$

where σ is the standard deviation, to attenuate the image and remove noise. A Laplacian operator

$$\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{2.3}$$

is applied to highlight regions of high-intensity changes. The two operators can be linearly combined as

$$\nabla^2 G(x,y) = \frac{x^2 + y^2 - 2\sigma^2}{\pi\sigma^4} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} \tag{2.4}$$

.

To detect a blob of radius s, a standard deviation of $\sigma = \frac{s-1}{3}$ is recommended, since 99% of a Gaussian is concentrated within 3σ [1]. Many approaches were proposed to detect blobs of various shapes in an image by varying the size of σ [1, 3].

One approach to identify failing rain streaks candidates in a rainy image is to first detect blobs, then select the ones that match certain criteria of rain streaks, including breadth-to-width ratio, orientation, and average blob intensity, relative to surrounding pixels. Figure 2-3 shows how blob detection can be used to identify rain streak candidates in a falling rain detection algorithm.

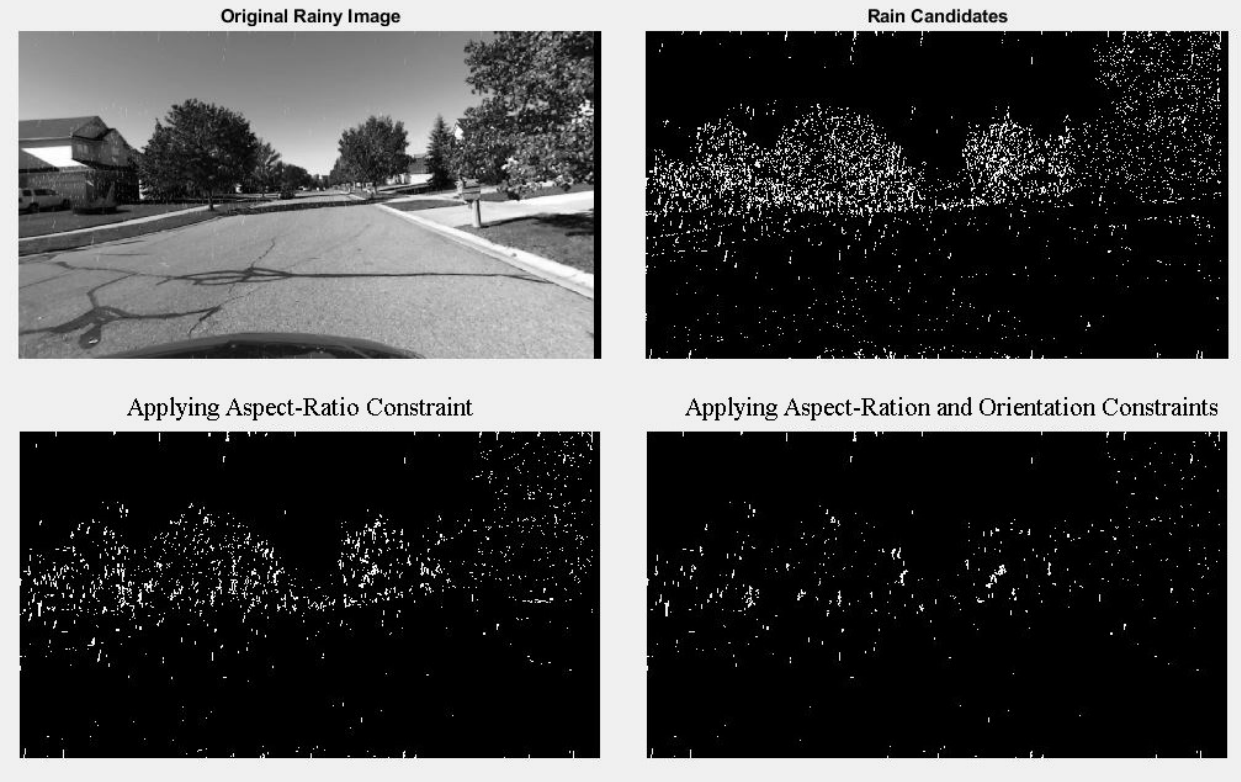


Figure 2-3: Starting with a rainy image (top left), blob detection is used to identify blobs in the image (top right). An ellipse that best contains the blobs is identified and the aspect ratio of its major-to-minor axes is used to eliminate non-rain streak blobs (bottom left). Ellipse orientation is finally used to eliminate more blobs that do not satisfy the rain streak orientation constraint (bottom right).

2.1.3 Depth Estimation

Depth estimation means estimating the distance of different objects in the environment, from an observer. The observer can be the human eye or any sensing mechanism that is sensitive to distance changes, including Radar, Lidar, and ultrasonic. Many applications in the automotive domain, such as 3-D mapping, navigation, and augmented reality use depth estimations as an integral part of their algorithms. Sensor fusion of different senor inputs, like camera and ultrasonic) is usually used, since it increases depth estimation accuracy and improves system reliability. For vision-based depth estimation, depth from stereo images is the most commonly-used approach, though good progress has been reported on depth estimation through deep-learning, using monocular images (e.g., see Godard et al. [4]). Stereo depth estimation is based

on measuring the location disparities of scene points, captured by stereo camera set. As shown in Figure 2-4, the same scene point *P* is projected at different positions on the image planes of the two-camera stereo system. Through simple trigonometry, we can show that the disparity $d = X_1 - X_2$ can be used to calculate the distance Z of point P from the cameras as

$$Z = \frac{f.b}{d} \tag{2.5}$$

where *f* is the camera focal length and *b* is the distance between the camera centers.

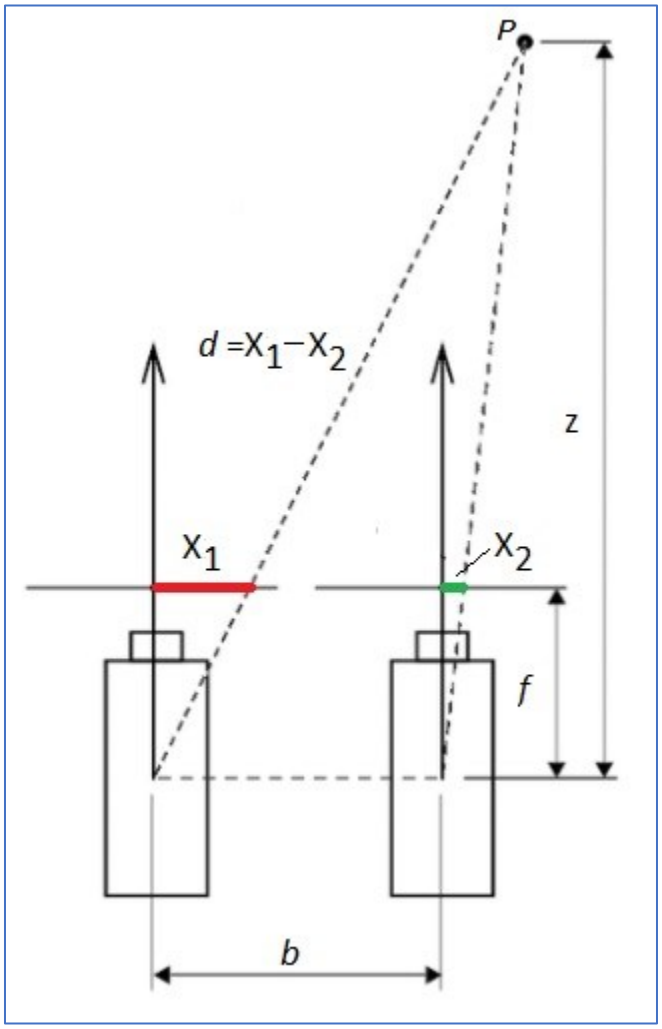


Figure 2-4: The position disparity of scene points as captured on stereo camera image planes can be used to estimate the distance to scene points. Distance is inversely related to disparity measure, meaning that further scene points exhibit smaller disparity than closer ones to the cameras

2.1.4 Brightness Adjustment

Brightness adjustment refers to the process of shifting the intensity of image pixels from the original level, to a more desirable one. This is an important step in vision algorithms that are sensitive to intensity changes. Feature matching and image restoration are two common image processing applications that may benefit from brightness adjustment. The brightness adjustment can be achieved by linearly shifting all image pixels with a fixed bias, or by nonlinearly shifting the brightness zone (minimum to maximum intensity range) to a different zone. Figure 2-5 shows different methods for applying the brightness adjustment process to an image. A fixed bias can be added to image pixel intensities, resulting in a brighter image with almost no change in contrast (top right). In the bottom left image, the histogram equalization technique was used to redistribute the original image intensity range over the whole [0-255] range. In the bottom right image, the original image intensity range was remapped to a new (tighter) range, resulting in a darker image with less contrast.

Figure 2-5: Some techniques to adjust the brightness of the image (top left) include adding a fixed bias (top right), using histogram equalization (bottom left), and intensity remapping (bottom right).

2.1.5 Barrel Effect Removal

Barrel or fisheye distortion is one of two common curvilinear distortions, caused by camera lenses, the other being pincushion distortion. The further the image points are from the center of the camera imaging sensor, the more curved inwards the image lines appear. Pincushion distortion is caused by low range or telescopic lenses and is opposite to barrel distortion, in a sense that the further the image points are from the center of the camera imaging sensor, the more curved outwards the image lines appear. For best performance, we ideally want to rectify distorted images, to get as close to a rectilinear image as possible. Figure 2-6 shows how barrel and pincushion distortions affect a rectilinear image.

Figure 2-6: The original image (top) is captured using rectilinear lenses. The fisheye image (middle) is captured using a captured with a wide-angle lens. Using a long-range lens causes pincushion distortion (bottom)

In automotive applications, barrel distortion is more common, since many vehicle onboard cameras use wide-angle lenses to capture as much as possible of the scene around the vehicle. Mathematically, barrel distortion is approximated as a single-parameter polynomial as [5]

$$\begin{aligned} x_u - x_0 &= (x_d - x_0)(1 + \lambda r_d^2) \\ y_u - y_o &= (y_d - y_o)(1 + \lambda r_d^2) \end{aligned} \tag{2.6}$$

Or equivalently:

$$\begin{aligned} r_u &= r_d(1 + \lambda r_d^2) \\ r_d &= \sqrt{(x_d - x_o)^2 + (y_d - y_o)^2} \\ r_u &= \sqrt{(x_u - x_o)^2 + (y_u - y_o)^2} \end{aligned} \tag{2.7}$$

where (x_o, y_o) are the coordinates of the image center, (x_u, y_u) are the coordinates of the undistorted image pixels, (x_d, y_d) are the coordinates of the pixels in the distorted image, r_d is the Euclidian distance from distorted image pixels to the image center, r_u is the Euclidian distance from undistorted image pixels to the image center, and λ is the distortion faction that is dependent on the type of lenses used.

2.1.6 Image Inpainting

Image inpainting refers to the collection of techniques used to replace certain sections of an image with other sections, either from the same image or from another one with similar characteristics. Inpainting is generally used to remove undesired elements of an image or reconstructing corrupt sections of an image, due to noise or occlusion. The most common Inpainting techniques are structural-based inpainting, texture-based inpainting, and exemplar-based inpainting [6, 7]. In structural-based inpainting, Partial differential equations are used to diffuse pixels from surrounding regions to the target region, preserving the direction of isophotes

(lines with constant light intensity). The process continues until all pixels are replaced. This technique produces a blurring effect that helps remove small defects in the image. It is not suitable, however, for large areas and to reconstruct regions with rich texture [7]. Texture-based inpainting is done by selecting patches from the source region and copying them to the missing sections of the target region until all target area is covered. Exemplar-based inpainting can be considered a hybrid inpainting technique that borrows the patch copy-and-paste from texture-based inpainting, and the diffusion approach from structure-based inpainting. The process starts by identifying the missing sections in the target region that requires inpainting. Fill priorities are then assigned to target pixels which determines the order in which these pixels are replaced. Next, patches that best match the target regions are selected and used to iteratively fill gaps in the target region. After each iteration, the target region boundary is updated, and fill priorities are updated [8]. In rain removal applications, image inpainting is used to restore areas of the rainy image that are corrupted or occluded by the rain presence. Figure 2-7 shows how image inpainting can be used to restore image areas corrupted with raindrops.

Figure 2-7: For a rainy image (top), the raindrop detection algorithm identifies areas of a raindrop in the image (middle). Image inpainting is then used to restore the image, effectively removing rain from the image (bottom)

2.2 Image Quality Metrics

This section includes a summary of some of the most common metrics used in literature for evaluating the quality of images. Wang et al. [9], classified Objective image quality metrics as full-reference**,** reduced-reference, or no-reference, depending on the availability of the original, distortion-free image or a set of extracted features, representing that image. Mean Squared Error (*MSE*), Peak Signal to Noise Ratio (*PSNR*), and Structural Similarity Index (*SSIM*) are examples of full-reference quality metrics. Blind/ Referenceless Image Spatial Quality Evaluator (*BRISQUE*) and Perception-based Image Quality Evaluator (*PIQE*) are examples of the no-reference quality metrics [10].

2.2.1 Mean Square Error

MSE measures the average of the square of errors in the estimation of reference values. Mathematically it is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \hat{X}_i\right)^2 \tag{2.8}$$

where X and $\hat{X}$ are the reference and estimated sets, respectively and n is the total number of estimations. Due to its simplicity, MSE is sometimes used as a cost function in deep-learning networks, instead of the more common ℓ_1 and ℓ_2 metrics (see for example Qian et al. [11]). Figure 2-8 shows one critical weakness of *MSE*, namely that it is not a good representation to image quality, as perceived by the human eye. Despite being much clearer to human perception, the speckle noise image scored worse in terms of image quality (higher MSE) than the almost unrecognized salt-and-pepper one.

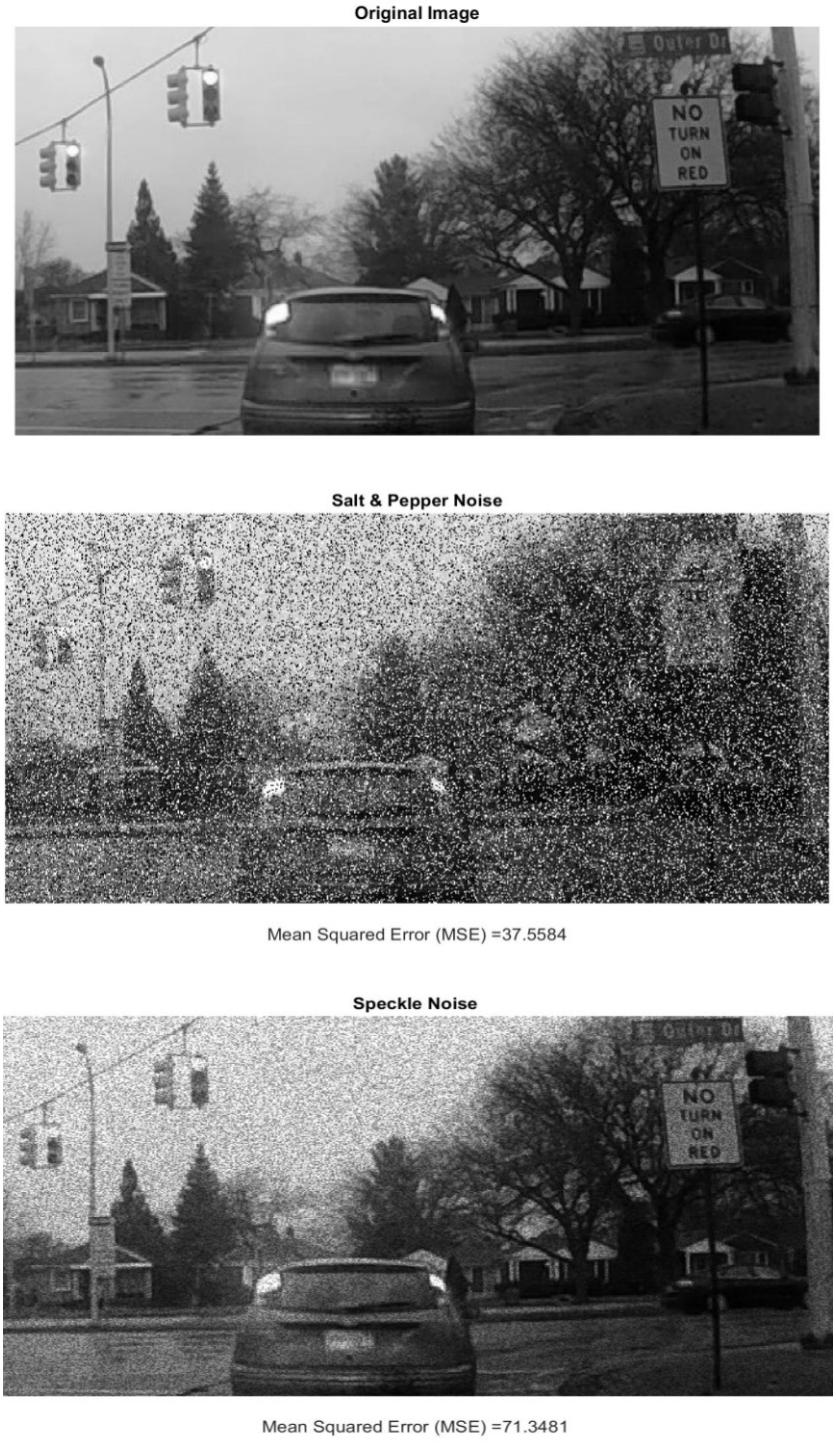


Figure 2-8: A Clear Image (top) is corrupted with 'salt and pepper' noise (middle) and 'speckle' noise (bottom).

2.2.2 Peak Signal-To-Noise Ratio

Peak signal-to-Noise Ratio (PSNR) is the ratio between the maximum power of a signal and the power of noise corrupting the signal and is usually expressed in logarithmic scale (base 10). PSNR can be expressed in terms of MSE as follows [12]:

$$PSNR = -10\log_{10}\left(\frac{MSE}{MAX_I^2}\right) = 20\log_{10}(MAX_I) - 10\log_{10}(MSE) \qquad (2.9)$$

where MAX_I is the maximum possible pixel value of the image and *PSNR* is given in dB. For an 8-bit image, $MAX_I = 255$, and *PSNR* can be given as $PSNR = 48.13 - 10\log_{10}(MSE)$.

This last formula shows that *PSNR* is strongly (inversely) related to *MSE*. It is no surprise that, like *MSE*, *PSNR* scores do not align well with human perception of image quality, as shown in Figure 2-9. One could argue that despite the blurriness, a human observer may recognize more of the image content in the Gaussian noise image, than the salt and pepper noise image. PSNR score for the two noisy images, however, does not align with human perception of image quality.

Figure 2-9: A Clear Image (top) is corrupted with 'salt and pepper' noise (middle) and ‘Gaussian’ noise (bottom).

2.2.3 Least Absolute Deviations and Least Squares Error

Least Absolute Deviations, also known as Least Absolute Error (*LAE*) is commonly used in Convolutional Neural Networks (CCN) as a loss function ($\ell_1$loss). The goal of a CNN is to minimize this error to convert to a satisfactory solution. It is a simple metric that can be given mathematically as

$$LAE = \sum_{i=1}^{n} |y_i - f(x_i)| \tag{2.10}$$

where y_i is the target (reference) value, $f(x_i)$ is the estimated value for an input x_i, and n is the number of samples.

Least Squares Error (*LSE*) is another metric commonly used as a loss function in CNN-based systems(ℓ_2 loss). Mathematically it is given as

$$LSE = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{2.11}$$

We notice that MSE differs from LSE only by the sample averaging factor $\frac{1}{n}$.

Both ℓ_1 and ℓ_2 are usually added as penalty terms to the regularization process in CNN-based algorithms which is designed to prevent overfitting during the training stage. The topic of using ℓ_1 versus ℓ_2 and the difference between the two loss functions has been discussed in many machine-learning blogs. The following Table 2-1 shows some of the differences outlined by N. Tyagi [13]

Table 2-1: Properties of ℓ_1 versus ℓ_2 loss functions

Loss function Attribute	ℓ_1	ℓ_2
Penalization method	Penalizes the sum of the absolute value of weights	penalizes the sum of square weights.
Sparsity	Has a sparse solution	Does not have a sparse solution
Robustness to outliers	Robust to outliers	Not Robust to outliers
Solution uniqueness	No unique solution	A Unique solution can be reached

2.2.4 Normalized Cross-Correlation

Normalized Cross-Correlation (NCC) is a (normalized) measure of similarity of two series, as a function of the displacement of one relative to the other [14]. In image processing, NCC is used mostly for template matching and image restoration. A segment in a reference image is considered a match to one in a target image if NCC between them is highest. Normalization is achieved by subtracting from each series its mean, then dividing the result by its standard deviation. This helps alleviate the effect of variations of illumination intensities or due to using different sensors to capture each image (e.g., in stereo-vision applications). mathematically, NCC can be given by:

$$NCC = \frac{1}{n\sigma_r\sigma_t}\sum_{i=1}^{n}(X_r - \mu_r)(X_t - \mu_t) \tag{2.12}$$

where n is the total number of pixels tested in each image, $X_{r,t}$ are the reference and target images, and $\mu_{r,t}\ and\ \sigma_{r,t}$ are their corresponding means and standard deviations, respectively.

2.2.5 Structural Similarity Index

The quality metrics disused so far are simple to calculate and understand but they do not reflect the human perceived visual quality. Wang et al. [9] presented an image example with different types of distortions applied to it. As we showed in Figure 2-8, the speckle-distorted image scored worse than the one distorted with salt and pepper noise in terms of image quality (per MSE metric), even though its perceptual quality was much better. The structural information of the reference image was preserved in the contrast-stretched image and could be recovered via linear transform which was not the case with other types of distortion. Natural image signals are highly structured and the human visual system is adapted to extract structural information from images. Structural similarity (SSIM) is an image quality assessment measure designed around the human visual system (HVS). It makes use of structural information change to provide an approximation to perceived image distortion. The SSIM metric is constructed as a combination of luminance, contrast, and structure comparators as follows [9]:

1- The luminance of each image signal is estimated using its mean intensity:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.13}$$

where μ_x is the mean intensity of signal x, N is the total number of pixels and x_i is the intensity at pixel i.

2- Luminance comparison function $l(x, y)$ is then given by:

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{2.14}$$

where the constant $C_1$is added to protect for the near-zero denominator.

3- The contrast of each image signal is approximated by its standard deviation $\sigma_{x,y}$:

$$\sigma_x = \left(\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)^2 \right)^{\frac{1}{2}} \tag{2.15}$$

4- Contrast comparison function $c(x, y)$ is then given by:

$$c(x, y) = \frac{2\sigma_x \sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{2.16}$$

where constant C_2 is added to protect for the near-zero denominator.

5- Image signal structure is associated with the unit vectors $\frac{x-\mu_x}{\sigma_x}$ and $\frac{y-\mu_y}{\sigma_{xy}}$

6- The structure comparison function is defined as:

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3}$$
$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_x)(y_i - \mu_y) \tag{2.17}$$

where C_3 is added to protect for the near-zero denominator.

7- Structure similarity index SSIM is then given by:

$$SSIM(x, y) = [l(x, y)]^{\alpha} . [c(x, y)]^{\beta} . [s(x, y)]^{\gamma} \tag{2.18}$$

Figure 2-10 shows how SSIM is more aligned to human perception compared to MSE and PSNR metrics.

Figure 2-10: When comparing an image to itself (top left), MSE =0, PSNR=infinity, and SSIM =1, as expected. MSE and PSNR scores, however, are not aligned to human perception of image quality, unlike SSIM.

2.2.6 Earth Movers Distance

In image processing, EMD reflects the cost of moving one image, represented by some feature signature (e.g., intensity histogram), to a reference image, represented with the same signature type. Given two images, Image1 and Image2, one or more features are selected and clustered, to create signatures, [15]

$$S = \{(s_1, w_{s1}), (s_2, w_{s2}), \dots, (s_3, w_{sm})\}$$
$$T = \{(t_1, w_{t1}), (t_2, w_{t2}), \dots, (t_m, w_{tm})\} \quad (2.19)$$

for Image1 and Image2, respectively.

s_i, w_{si} for $1 \le i \le m$ and t_j, w_{tj} for $1 \le j \le n$ represent the cluster Ids and weights for the two signatures. A feasible flow $F = [f_{i,j}]$ between the two signatures must satisfy the following constraints:

$$\begin{gathered} f_{i,j} \ge 0, \forall\, 1 \le i \le m, 1 \le j \le n \\ \sum_{i=1}^{m} f_{i,j} \le w_{si}, 1 \le i \le m \\ \sum_{j=1}^{n} f_{i,j} \le w_{tj}, 1 \le j \le n \end{gathered} \tag{2.20}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j} = min\left\{\sum_{i=1}^{m} w_{si}\,, \sum_{j=1}^{n} w_{tj}\right\}$$

Given a ground distance $D = [d_{i,j}]$ between the two clusters, EMD represents the solution that minimizes the work (flow times distance) to move one signature to match the other. Mathematically, this is given by [15]

$$EMD(S,T) = min \sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}\, d_{i,j} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}\, d_{i,j}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{i,j}} \tag{2.21}$$

Figure 2-11 shows how the *EMD* metric represents the cost of restoring a progressively degraded image, corrupted with different types of noise. One issue with the *EMD* metric is that there is no clear correlation between human perceived image quality and *EMD* score. Salt and pepper noise image series generally had lower EMD scores, though perceptually they are worse than the Gaussian-degraded images. Based on this, the EMD metric is maybe suitable to track the progression of image quality in, for example, a learning algorithm from one integration to the next. It may not, however, be suitable for assessing the degradation levels of different images, compared to a reference.

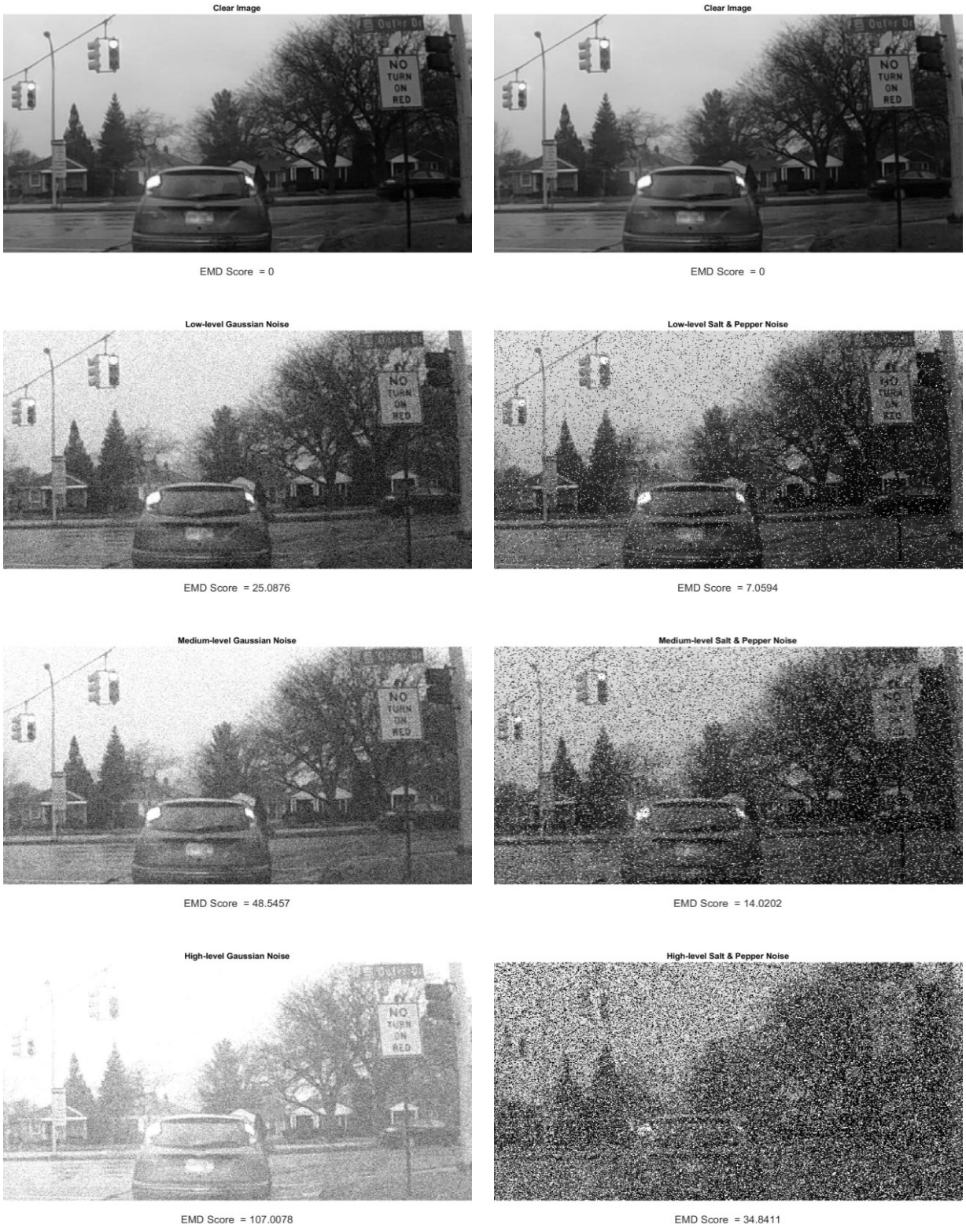


Figure 2-11: The EMD metric can be used to track the progression of image quality with increased/decreased noise in a series (top left to bottom left or top right to bottom right). It is not, however, used the measure image degradation between different images qualitatively (left to right).

2.3 Model Performance Metrics

Regression and classification are the two main problems that machine learning (including deep-learning) models are designed to solve. A regression model maps an input x to an out y through a specific mapping function. Mathematically, this is given by

$$y_i = f(x_i, \beta) + e_i \tag{2.22}$$

where β represents a set of unknown parameters and e_i is an additive error term.

MSE, RMSE (Root-MSE), and MAE are some commonly-used metrics to evaluate the performance of a regression model.

Classification models, on the other hand, try to answer the question: to which class of available classes does a sample input most likely belong? The classification can have one of the following four outcomes:

1. Sample input is correctly classified to belong to the specific class. This represents a True Positive (TP).
2. Sample input is correctly classified as not belonging to a specific class. This represents a True Negative (TN).
3. Sample input is incorrectly classified to belong to a specific class. This represents a False Positive (FP).
4. Sample input is incorrectly classified as not belonging to a specific class. This represents a False Positive (TP).

Accuracy, Precision, Recall, Jaccard index, and Dice Metrics are designed to evaluate classification performance, based on the outcome of classifying inputs samples.

2.3.1 Precision Metric

Precision is also called positive predictive values (PPV) and it represents the portion of positive results that are true positive. It is given by

$$Precision = \frac{TP}{TP + FP} \tag{2.23}$$

The precision metric is desirable when the cost of false positives is high in the classification model. In an automotive predictive brake system, for example, too many false classifications of obstacles in the path of the ego vehicle may result in the excessive application of the braking system and reduction in brakes' expected lifetime.

2.3.2 Recall Metric

The recall is also known as Sensitivity and it represents the portion of positive results that are correctly predicted positive. It is given by

$$Recall = \frac{TP}{TP + FN} \tag{2.24}$$

The recall is desirable when the cost of false negatives is high in the classification model. For the same braking system described above, misclassifying an imminent collision scenario as a no-collision event may cause a crash event that is prohibitively expensive in terms of safety. This example explains why the Recall and Precision metrics are rarely considered in isolation. It is worth noting that both Precision and recall provide some information on the rate and type of classification error but both ignore the negative cases since TN outcomes are not part of the calculation.

2.3.3 Accuracy Metric

The accuracy of a classifier system is an indication of overall classification performance. It is the ratio of correct predictions (TP or TN) to all prediction outcomes. It is given by,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.25}$$

2.3.4 Dice Coefficient

Sørensen–Dice coefficient or F_1 score is a similarity metric that represents the harmonic mean of Precision and Recall metrics [16]. Mathematically, it is given by

$$Dice = \frac{2|target \cap prediction|}{|target| + |prediction|} \tag{2.26}$$

where $|x|$ represent the cardinality (number of elements) of set x. For Boolean data, this metric can be given by [16],

$$Dice = \frac{2TP}{2TP + FP + FN} \tag{2.27}$$

2.3.5 Jaccard Index

Jaccard index, also known as Intersection over Union (IoU), is another similarity metric that is closely related to Dice. It is given by [17],

$$IoU = \frac{target \cap prediction}{target \cup prediction} \tag{2.28}$$

Or for Boolean data,

$$IoU = \frac{TP}{TP + FP + FN} \tag{2.29}$$

From which the relation between Jaccard index and Dice can be given by [18]:

$$\begin{aligned} Dice &= \frac{2IoU}{IoU + 1} \\ IoU &= \frac{Dice}{2 - Dice} \end{aligned} \tag{2.30}$$

Though the two metrics are almost functionally equivalent, IoU scores tend to be closer to worst-case performance, whereas Dice scores are closer to average performance [19]. IoU score is used in many classification applications as a predefined threshold to determine whether any given prediction is considered a TP or an FP. This threshold is sometimes referred to as the penalty threshold. In an object threshold algorithm, for example, setting the threshold to an IoU=0.5 means that at least 50% of the pixels in the predicted object must match those of the ground truth object, for the detection outcome to be accepted (TP). Figure 2-12 and shows the results of a binary classifier that attempts to classify sample points as either belonging to the diamond-shaped area or not. Table 2-2 shows the results of calculating accuracy, precision, recall, dice, and IoU metrics for the classification results.

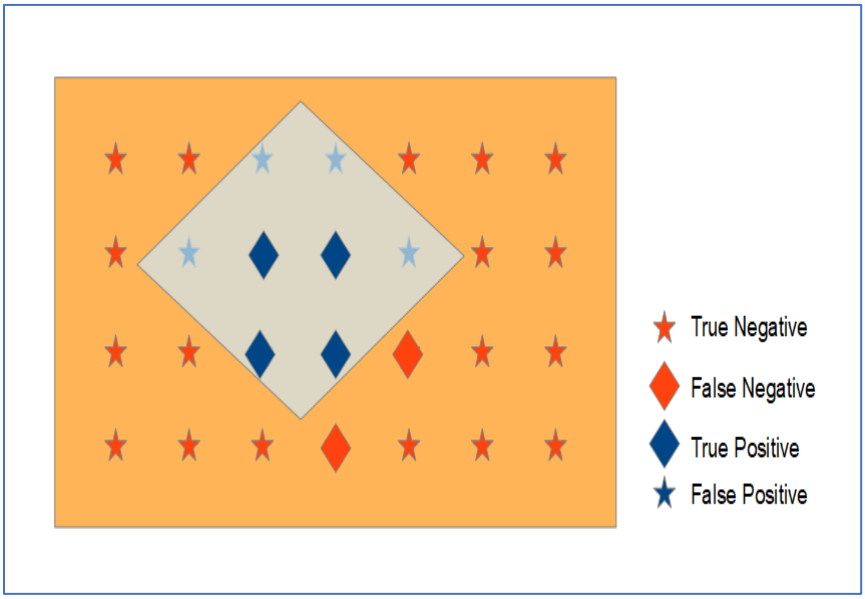


Figure 2-12: A binary classifier tries to determine if a sample belongs to the diamond shape or not.

Table 2-2: Similarity metric results for the binary classifier example.

Variable/ Method	TP	TN	FP	FN	Precision	Recall	Accuracy	IoU	Dice
Measurement/ calculation	4	18	4	2	0.5	0.667	0.786	0.4	0.571

2.3.6 Average Precision (AP)

One common technique in the evaluation of the performance of classification models is to plot the precision-recall curve, for a given classification confidence level threshold (penalty threshold) as mentioned earlier, IoU is commonly used to represent this penalty threshold. This curve has an average negative slope since for a given penalty threshold, precision tends to decrease as recall increases. This curve is useful in representing the tradeoffs of precision versus recall for a given classification model.

AP is one way to capture the characteristics of the precision-recall curve in one metric and is calculated as the Area Under the Curve (AUC) of the function representing precision as a function of recall. The PASCAL VOC2011 [20] challenge slightly modified how AP is calculated, to eliminate the oscillatory pattern in the precision-recall curve. In this version, the precision for recall r is set to the maximum precision calculated for any $r' \geq r$. This method simplifies the AP calculation since the new AUC is now a collection of rectangles with easily calculated areas. Figure 2-13 shows an example of a precision-recall curve and the AUC as defined in the original and the modified AP calculations. In either method of calculation, a higher AP score is associated with the better overall performance of the classification algorithm, at a given penalty threshold. A family of precision-recall curves can be plotted on the same figure, each representing a class in the classifier. A set of curves can also be plotted, representing the precision-recall relations at different penalty thresholds. These two plots are related to the

way the Mean Average Precision (mAP) is calculated, which is not standard across vision challenges. In the PASCAL VOC2011 [20] challenge only one penalty threshold is used at IoU=0.5 to generate the precision-recall curves for all classes. mAP, in this case, is the average of the AP scores across all classes. The COCO [21] challenge, on the other hand, calculates the AP at penalty threshold IoU range from 0.50 to 0.95, at 0.05 resolution, for each class. The mean AP is then calculated per class and the mAP is the average of these means over all classes.

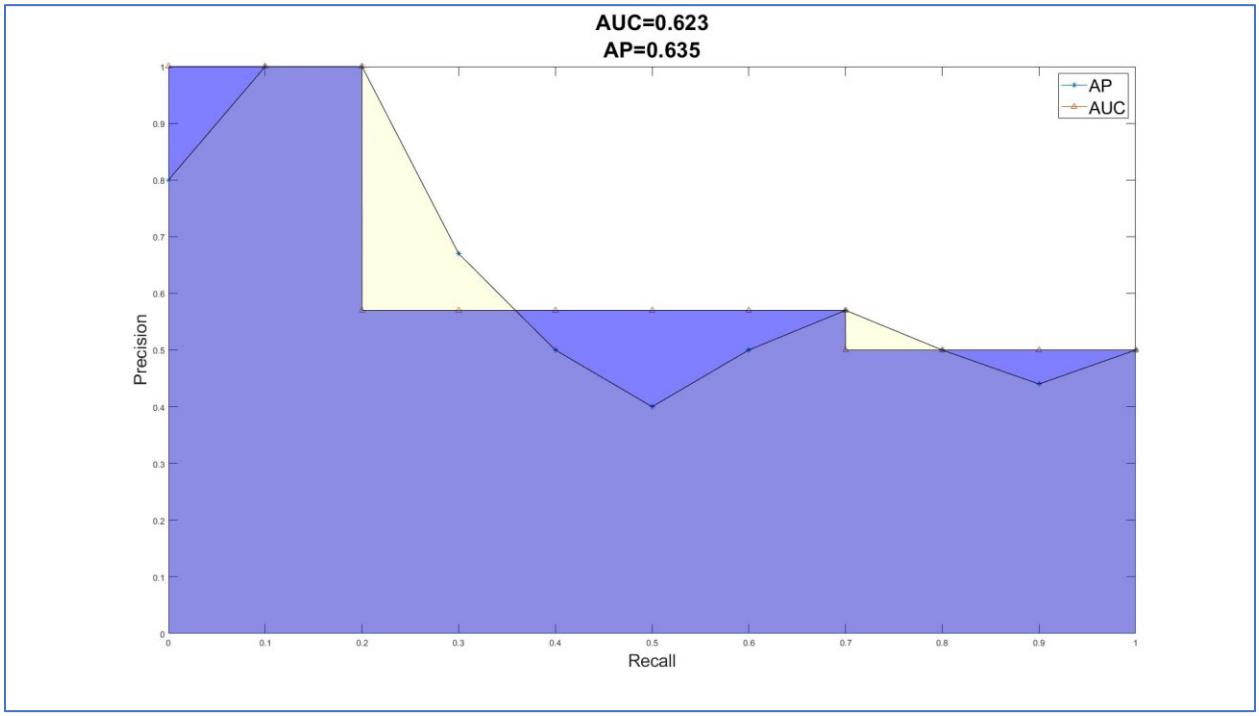


Figure 2-13: AP score was originally calculated as simply the Area Under the Curve that represents the precision as a function of the recall. To eliminate the fluctuations in the precision values, the precision at any point with recall r is set to the maximum precision value of points to the right of it (r'>=r). As shown in the plot above, the (new) AP score is different from the original AP score that was calculated as the AUC.

Chapter 3

Rain Detection and Removal Techniques

3.1 Introduction

Rain, a source of adverse weather conditions, deteriorates the quality of the images and negatively affects the performance of vision-based algorithms which use these images as the main source of information [22] [23] [24]. Rain can be presented in an image in two forms, falling rain streaks and adherent raindrops that accumulate on the camera lens cover or the vehicle windshield. The characteristics of rain in its two forms are different, and the distortion it introduces to the image is accordingly different. Because of this, researchers tackle the rain-induced issue as two separate problems, and the solution to one form of rain distortion does not usually work well for the other form. The image quality metrics according to Wang et al. [9], can be classified as no-reference, reduced reference, or full-reference, based on the availability of original (distortion-free) images for evaluation. In the context of rained images, distortion and image quality refer to the quality of the rained images as compared to the rain-free ones. Rain introduces irreversible degradations to the image by distorting its pixels, modifying some of the image characteristics, and occluding certain objects in the image background that are of interest to the target vision-based algorithms. The falling rain is both uncontrollable and unpredictable which makes it impossible to construct a reference (ground truth) dataset for reconstructing a rain-free image from a rained one. In that sense, rained images fall under the no-reference category based on Wang et al.'s [9] definition. Image restoration techniques try to make use of

certain features in the rain-free images, such as brightness and texture, to recover the sections of rained images that are distorted by rain. This brings the complexity of image restoration from the no-reference level to the reduced reference one. Attempts have been proposed to use synthetically-generated rain that is added to rain-free images, to create perfectly matched rain-free/ rained datasets. This brings the image restoration problem closer to the full-referenced level of complexity, but it remains highly dependable on the quality of the rain simulator algorithm used in the process.

We published a survey paper on the different approaches for adherent raindrop modeling, detection, and removal [25]. The survey described algorithms based on machine learning, as well as deep learning techniques.

3.2 Summary

In this section, we summarize in tabular format, the most important aspects of the adherent raindrop removal systems that were described our survey paper [25]. Table 3-1 lists the different raindrop models that were described in this paper. Table 3-2 summarizes the common classical approaches for raindrop detection. A list of different de-raining techniques is shown in Tables 3-3. Table 3-4 shows a comparison of the Deep Learning and CNN approaches to image de-raining that were described in that survey paper.

Table 3-1: Raindrop models.

ID	Basic Idea	Potential limitations
Cord et al. [26]	Assume elliptical shape for raindrops and use axes aspect ratio, size, and brightness constraints as a model for raindrops.	It May not account for irregular raindrop shapes and the effect of background texture on raindrop appearance.
Kurihata et al. [27]	Used a PCA algorithm to generate eigendrop templates	Does not account for the effect of texture on raindrop appearance.
Fouad et al. [28]	Use a declivity operator to describe raindrops as a sequence of peaks and valleys.	Do not consider the background composition's role in the appearance of raindrops.
Halimeh et al. [29]	Developed a complex model (RIGSEC) for a raindrop, based on its geometric and photometric properties.	Assuming part of a sphere for a raindrop and ignoring the blurring effect of a raindrop may limit model accuracy.
Roser et al. [30]	Added blurriness effect to RIGSEC and limited the rendering of raindrop models to certain regions of the image to reduce rendering time.	Generating raindrop models at specific regions in the image may lower the rate of matching with real raindrops.
Sugimoto et al. [31]	Used MSER to improve the initial detection of potential raindrops and spheroid for raindrop approximation.	Added complexity may make the model less appropriate for real-time applications.
Stuppacher et al. [32]	Modeled raindrops using height maps, considering raindrop dynamics and water content losses and gains for moving raindrops.	The model is more suitable for CGI applications to generate realistic raindrop effects.
Roser et al. [33]	Modeled raindrops using Bézier Curves.	Reliance on approximations of raindrop size from correlations between 2D ratios and tilt angles reduces model accuracy.

Table 3-2: Raindrop detection - classical approaches.

ID	Application	Approach	Potential limitations
Yan et al. [34]	Weather classification in the automotive domain	Use AdaBoost to combine two weak classifiers, HGA and HSV. Classifies weather as Rainy, Cloudy or Sunny	Applications of weather classifiers are limited in the automotive domain to ADAS warnings and windshield wiper triggers.
Wu et al. [35]	Raindrop detection in the automotive domain	Use AdaBoost to combine color, shape, and texture saliency maps. Create a raindrop visual descriptor and use SVM to classify the weather.	Assumes circular 2D shape of a raindrop and fails under heavy rain conditions
Liao et al. [36]	Raindrop detection in the automotive domain	Segment the scene into the roadway and building areas. Identify raindrop candidates through edge detection and binarization and compare their dimensions to the closest ellipse.	The detection algorithm might be slow for real-time automotive applications and it fails to handle background noise and large raindrops.
Ishizuka et al. [37]	Raindrop detection in the automotive domain	Daytime Detector uses Sobel for initial identification, then texture	The optical flow approach used assumes straight-line driving and may fail on winding roads. It may also cause incorrect classification

ID	Application	Approach	Potential limitations
		information, and optical flow to detect real raindrop pixels. Nighttime Detector eliminates light source pixels, then uses a temporal intensity change feature to identify raindrop pixels.	as raindrops, objects that are moving at the same speed as the test vehicle. (e.g. other vehicles).
Kurihata et al. [27]	Raindrop detection in the automotive domain	Use similarity degree between potential raindrops and eigendrop template to identify raindrop regions	Does not account for the effect of background variations on raindrop characteristics (texture, brightness).
Yamashita et al. [38]	Raindrop detection in surveillance applications	Match images from different cameras, then analyze intensity variance under low and high texture image backgrounds to detect raindrops.	Requires multiple cameras which reduces the common FOV, and assumes raindrops do not occlude the same section of the restored image.
Yamashita et al. [39]	Raindrop detection in surveillance applications	Capture successive image frames and identify them as raindrop segments, those that are detected near the expected location and satisfy size ratio constraints.	Requires precise knowledge of rotation angle and assumes idle raindrops between frames which is true only under light rain conditions.
Yamashita et al. [40]	Raindrop detection in surveillance applications	Similar to [39] but rotation angle is estimated and raindrop decision is made on a pixel base, by measuring the noise existence rates in the original and rotated image.	Assumes idle raindrops between frames which is true only under light rain conditions.
Yamashita et al. [41]	Raindrop detection in surveillance applications	Match stereo image pixels using NCC and apply one-on-one matching to eliminate noise. Compare measured to the expected disparity of raindrops to determine true raindrops.	Raindrops are blurry and may not result in good disparity measurements. Also, the long computational time is observed as a result of pixel-based calculations.
Yamashita et al [40]	Raindrop detection in surveillance applications	Create a compound image from the temporal image sequence and select raindrop pixels that show “often” in the noise candidate trajectory curve.	Requires many frames and involves many pixel projections.
Roser et al. [42]	Weather classification in the automotive domain	Use feature histogram to create a bag of features, and use SVM to classify weather as Clear, Light rain, or Heavy rain.	Relatively slow, due to the large descriptor. Error rate increases with background complexity increase.
Cord et al. [43]	Weather classification in the automotive domain	Compare the intensity gradient image to the threshold image and pick the strongest candidates. Pick raindrop regions based on dimensions and eccentricity constraints and through temporal analysis.	System Requires focused raindrops (camera needs to be attached far away from the windshield). Raindrop size is relatively small (3-10 pixels) which may cause reduced accuracy.

ID	Application	Approach	Potential limitations
Cord et al. [26]	Raindrop detection in the automotive domain	Segment the image then uses either watershed or background subtraction to identify potential drops. Use size, shape, and temporal constraints to identify real raindrops.	The algorithm runs slow due to implementation in MATLAB. Adding more frames improves performance but adds delay to the overall system operation time.
Nashashibi et al. [44]	Raindrop detection in the automotive domain	Detect potential raindrops through temporal intensity change and shape roundness. Use a lack of clear contour as a raindrop characteristic, then verify selection by spatially matching raindrop regions in consecutive frames.	Detection of unfocused is challenging and the algorithm fails under bright background conditions.

Table 3-3: Raindrop-degraded image recovery.

ID	Approach	Potential limitations
Liao et al. [36]	**For buildings ROI, replace the raindrop area with the closest non-rain area (using an 8-connected area template). For road ROI, use inpainting or morphological operations.**	**Removal time is long (0.44 to 0.68 seconds per frame) and it is proportional to rain density. The restoration of the road mark sections of the image is not perfect, due to the limitations of the inpainting method.**
Wu et al. [35]	Use the inpainting technique through smooth propagation in the equal intensity line direction.	Limited to low and medium rain intensity. Inpainting based on intensity does not preserve or recover the textural characteristics of the recovered regions.
Yamashita et al. [39]	Create a composite image of the original and rotated one, with a parameter that controls how much each image is contributing to the final composite one.	Chromatic variations between original and rotated images may still exist, even with correction. This affects the quality of the recovered image. The algorithm fails if the difference between original and rotated images is large.
Yamashita et al. [40]	Decompose the image into structure and texture images. Apply inpainting process on structure image and texture synthesis process on the other.	The Spatio-temporal analysis may be needed to improve texture recovery but this, in turn, may add delay to the processing time.
Yamashita et al [41]	Use disparity information to identify proper regions from the complementary image in the pair for raindrop pixel substitution.	Relies on imperfect disparity map data to select substation pixels. Also, the approach fails if raindrop noise is present in both images.
Roser et al. [30]	Estimate translational and rotational parameter vector h probabilistically, then use multi-band blending to recover rained regions.	While producing good results, this algorithm, both in its detection and recovery section seems to be too computationally expensive for automotive applications.

Table 3-4: Deep learning and CNN approach to image de-raining.

Raindrop Removal System	**Network Architecture**	**Datasets and Testing**	**Potential Limitations**
Dirt and Rain Noise Removal (Eigen et al. [45])	multilayer convolutional network with two hidden layers with 512 units each.	Pictures of a glass plate with dirt and water drops were taken. Patches of size 64 X64 for dirt and were paired with clear patches and used to train and test the rain and dirt remover system.	Restored images showed visible artifacts and were blurred where the raindrop/ dirt particle was removed.
Attention GAN Raindrop Removal Algorithm (Qian et al. [11])	1. Generative Network: a. Attention Map (3 layers of ResNet + 1 LSTM) b. Autoencoder (16 conv-ReLU with skip connection). 2. Discriminative Network: 7 convolution layers with the kernel size (3 x 3), a fully connected layer of 1024 neurons, and a single neuron with a sigmoid activation function	1119 pairs of images (rainy and clear), with various background scenes and raindrops. Raindrops are synthesized by spraying water on a glass plate.	Limited dataset for training and testing. Need for raindrop mask for supervised learning
Joint Shape-Channel Attention GAN Raindrop Removal Algorithm (Quan et al. [46])	GAN-based on encoder-decoder architecture with ResBlocks in between and short and long skip connections. Joint physical and channel attention blocks	Used Qian et al. [11] dataset for training and testing. Uses PSNR and SSIM were used for evaluation and benchmarking.	Dataset limitations for training and testing. PSNR and SSIM scores were only marginally better than other algorithms evaluated.
Improved Raindrop Removal with Synthetic Raindrop Supervised Learning (Hoa et al. [47])	The system consists of three sub-networks rain detection network, (I-CNNN with 5 Conv layers + BN and ReLU activation, and 6 Resblocks reconstruction network, same as detection network with 8 ResBlocks refine network, 2 Conv layers, and 2 ResBlocks	Synthesized rainy images and used them for training and testing. Adam optimizer [48]was used for setting up training parameters. PSNR and SSIM were used for evaluation and benchmarking.	The quality of images generated by the Rain Synthesized needs more independent evaluation against real rainy images.
Raindrop-Aware GAN for Coastal Video	Encoder/decoder architecture with short and long skip connections.	Used Qian et al. [11] dataset as well as Anmok beach videos and paired image sets for training and	Though outperforming other methods in the coastal setup, for urban setup no clear

Raindrop Removal System	Network Architecture	Datasets and Testing	Potential Limitations
Enhancement (Kim et al. [49])		testing. PSNR, NIQE [50], and SSIM were used for evaluation and benchmarking.	improvement was observed over Qian et al. [11].
Self-Supervised Attention Maps with Spatio-Temporal GAN (Alletto et al. [51])	The system is made of two stages. 1. Single-Image Removal, with location map estimator and raindrop remover networks. both constructed based on encoder/decoder architecture. 2. Spatio-temporal Raindrop Removal, flow estimator provides optical flow is learned from previous frames and concatenated with rainy image and estimated location map in a decoder/ encoder GAN architecture.	Used Qian et al. [11] dataset as well as data set of augmented videos from DR(eye)VE dataset [52] with synthetically-generated raindrops. PSNR and SSIM were used for evaluation and benchmarking.	The quality of images generated by the Rain Synthesized needs more independent evaluation against real rainy images.
Concurrent channel-spatial attention and long-short skip connections (Peng et al. [53])	The system was built on the encoder/ decoder architecture, with channel and spatial attention blocks added. Short and long connections were also introduced.	Used Qian et al. [11] dataset for training and testing. Uses PSNR and SSIM were used for evaluation and benchmarking.	Dataset limitations for training and testing. The approach is similar to Quan et al. [46] with differences in network architecture. Would be interesting to compare the performance of one against the other.
Separation-Restoration-Fusion Network for Image Raindrop Removal (Ren et al. [54])	the system consists of three modules. 1. Region separation module was implemented as an image pipeline with classical techniques 2. Region restoration module MFGAN built on pyramid topology was used. 3. region Fusion module IODNet connection network using DenseASPP [55] was used to construct fusing module	Used Qian et al. [11] dataset for training and testing. Uses PSNR and SSIM were used for evaluation and benchmarking.	Images need preprocessing to classify regions of the image based on the severity of the raindrop. The classification was based on experimental results from a limited dataset and may not apply to other scenarios.

3.3 Conclusion

We described a range of research works in the field of adherent raindrop detection and removal, with a focus on applications in the automotive domain. Based on the reviewed research work in this paper, we conclude the following:

1- Adherent raindrop detection and removal is a more challenging problem than falling rain detection and removal, due to the persistence of adherent raindrops over many image frames and the irregularity of raindrop shapes and sizes.

2- Due to the closeness to the image plane, adherent raindrops look blurry and occlude larger areas of the captured image.

3- Due to the above, most reviewed algorithms performed poorly under heavy rain conditions or fast-changing scenes with many moving objects.

4- Simple detection algorithms were based on observed optical or physical characteristics of adherent raindrops and performed well if the presumed conditions were met. Performance is degraded quickly for any deviation from these conditions, including change of background image texture or illumination and the introduction of moving objects in the scene background.

5- Complex detection algorithms performed very well under low and medium rain conditions. The added complexity, however, can introduce unacceptable latencies in real-time applications for processing rained images and removing adherent rain.

6- Compromises were discussed to improve processing time that included limiting the ROI, reducing the number of model templates, and dimension reduction, among other things.

7- The application of Deep-learning and CNN seems to be a very promising approach for solving the raindrop detection and rain removal problems.

8- The use of PSNR and SSIM metrics may not be the best choice for performance evaluation and benchmarking among different CNN-based algorithms. Results reported by different researchers showed marginal improvement in PSNR and SSIM scores which may very much be within the statistical margin of error.

Chapter 4

Framework For Simulating and Removing Rain in Stereo-Image Videos

4.1 Introduction

Raindrops fall at high velocities relative to the exposure time of the camera, producing severely motion-blurred streaks in images. Fluctuations in image intensity that are caused by rain "can severely degrade the performance of a wide range of outdoor vision algorithms, including, feature detection, stereo correspondence, tracking, segmentation, and object recognition." [22] Some work was done to study the effect of rain and de-raining on the performance of vision-based systems [23] [56] but their work was limited to certain aspects of vision-based systems (e.g. feature extraction and feature tracking). Controllability of test environment and repeatability of test results are integral requirements to any scientific experiment. Falling rain is uncontrollable in nature and it is costly to simulate in a dynamic setup which may include a moving vehicle and moving objects in the background. It is also prohibitively costly to create repeatable drive cycles under clear and rainy conditions. As a result, any image quality metrics for vision-based systems under rainy and de-rained conditions will fall under the no-reference class, as per Wang et al. [9] classification. For research purposes, however, this obstacle can be overcome by one of the following approaches:

A. Capture images of videos for a static background under rainy and clear conditions. This approach is simple enough and allows the researcher to focus on studying rain effects in an

image or video. It is, however, not suitable for dynamic scenes, where either the camera, background elements or both are dynamically moving.

B. Capture images or videos in a dynamic, but controlled environment. This allows for studying more broad scenarios than the static background version but it is more complex and expensive to set up the test environment.

C. Add rain effect artificially to recorded videos or captured images. This technique provides maximum flexibility and control over rain variables such as speed, density, orientation, etc. The drawback is that added rain may not exactly be true rain in an image scene.

In this section, we present a stereo-based rain simulation and rain removal framework that facilities the study of the effect of rain and de-raining on vision-based automotive applications. Our work [57] is built on previous work presented by S. Starik et al. [58] and P. Barnum et al. [56] for rain simulation and rain removal techniques.

4.2 Rain Simulation Model

4.2.1 Method

We built our rain simulation model based on the following characteristics of falling rain in an image.

i. Falling Raindrops appear as rain streaks, due to the falling speed of raindrops and the exposure time per image frame. As described in section 3.2, Garg and Nayar [59] explained that a falling raindrop can occlude multiple image pixels on its way down, during each shutter exposure period.

ii. Rain streak pixels are generally brighter than their surrounding background pixels. This feature was attributed by Garg and Nayar [59] to the fact that raindrops act as tiny lenses, collecting light beams from their surroundings.

iii. The closer the rain streaks to the camera are, the longer they appear on the captured image. As shown in Figure 4-1, the mapping of any environment point $P(X, Y)$ and the image pixel $p(x, y)$ that represents it is a function of the lens focal length f, and the distance from the image plane z.

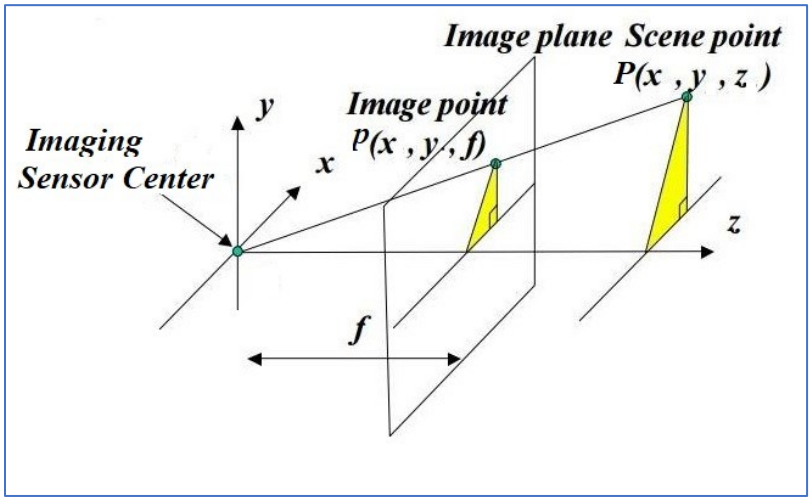


Figure 4-1: Ideally, the size of an object projection in an image is inversely proportional to the distance of that object from the camera image plane

For a fixed focal length, the magnitude of the pixel's x and y is inversely proportional to the distance z, as given in the equations below

$$x = f\frac{X}{z}$$
$$y = f\frac{Y}{z} \quad (4.1)$$

iv. The further the rain streaks from the camera are, the higher their density in the image becomes. This is also related to the projection of environment points onto image pixels. A far-away plane in the environment is projected into a smaller section of the image, as

compared to a closer plane. in other words, for the same real estate section in the image, more pixels are captured that represent further objects than closer ones from the image plane of the camera.

Based on these characteristics, we developed our falling rain simulator as follows:

A. Capture a set of images for scenes that we want to add falling rain into, using a stereo camera. We used the Zed" stereo-vision camera [60] that was mounted on the roof of a sedan car to capture videos of drives around the University of Michigan, Dearborn campus, at 20Hz rate and 720p resolution. We captured datasets for drive cycles during different weather conditions including clear and overcast. Figure 4-2 shows examples of some images we captured in these datasets.

(a) Left Image Clear | (b) Right Image Clear

(c) Left Image Overcast | (d) Right Image Overcast

Figure 4-2: Left/ Right image pairs captured with the Zed camera under clear and overcast weather conditions.

B. Calculate disparity maps from each stereo image pair and estimate the distance of objects from the camera accordingly. Disparity refers to the difference in the matching pixel locations, present in the left and right images of the stereo image pair. This process involves the following steps,

i. Find the matching pixels in the left and right images. Any matching algorithm can be used in this step. In MATLAB, the disparity function provides a choice between block-matching and semi-global block matching. In our implementation, we used the semi-global block matching since, as shown in figure 4-3, gave a smoother and more continuous disparity display, compared to the block matching one.

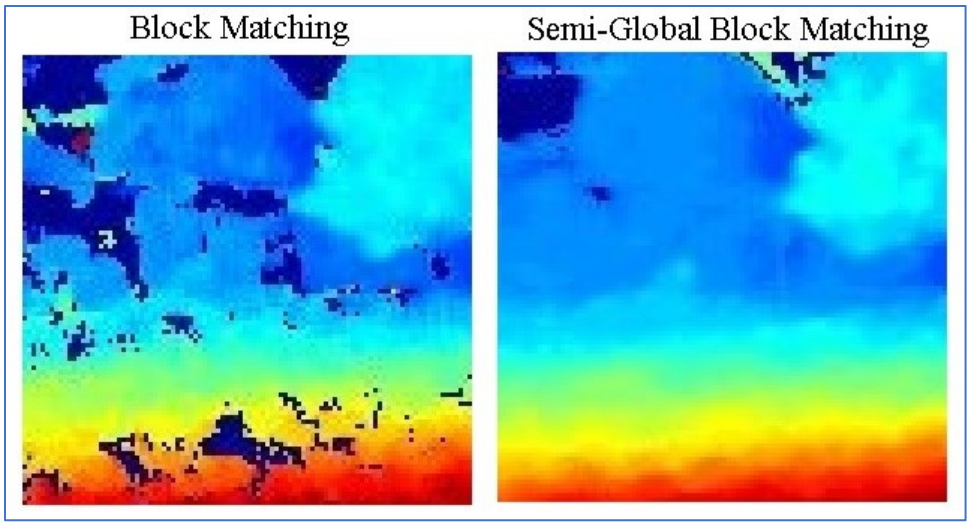


Figure 4-3: Semi-Global matching generates smoother, continuous disparity maps, as compared to the Block matching technique.

ii. Select a reference image (e.g., right image) and then rectify the other image, such that the corresponding pixels are located on the same rows of the reference image.

iii. Calculate the disparity D according to the equation

$$D = x - x' = \frac{Bf}{Z} \tag{4.2}$$

where x and x' are the distances of the matched pixels in the left and right image from the epipolar plane, and B is the distance (Baseline) between the center of the left and right

imaging sensor. As shown in Figure 4-4, the epipolar plane is the plane connecting the two imaging sensor centers to scene point X.

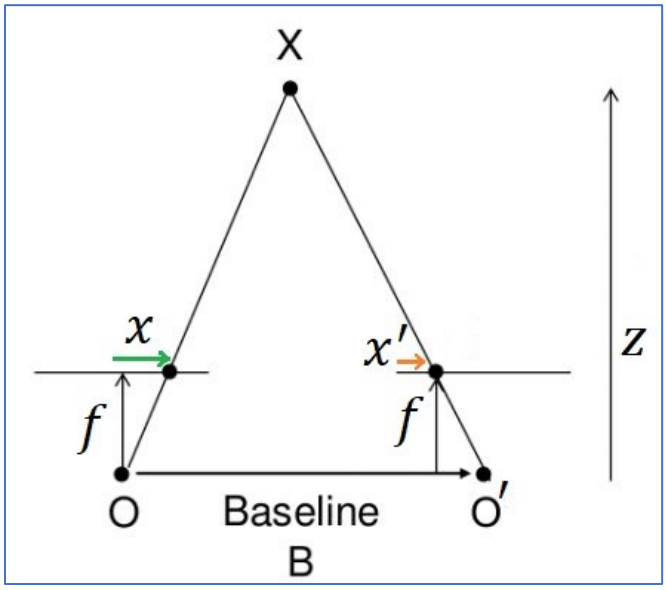


Figure 4-4: Disparity is calculated as the difference between distances of the matching pixels in the left and right images from the epipoler plane. The epipoler plane is the plane connecting the left and right image origins (O, O') and scene point X.

iv. Estimate the distance of image objects from the camera, based on their disparity values. Pixels of close objects have higher disparity values than pixels of further objects. For our application, we used relative distance from the camera which has an inverse relation to the disparity score.

v. Matching algorithms look for interesting features in the images, to match pixels in one image to those in the other. Some sections of the image, however, do not have sufficient features for matching, due to lack of gradient changes in the image characteristics (e.g., brightness), that translate to features of interest. Examples of these feature-poor segments may include the blue sky and the gray, unmarked road.

If the pixels cannot be matched, the disparity cannot be calculated. To rectify this issue, we employed a simple technique that proved effective in filling the missing disparity gaps. We assumed that any areas with missing disparities above a certain level (less than a y-axis

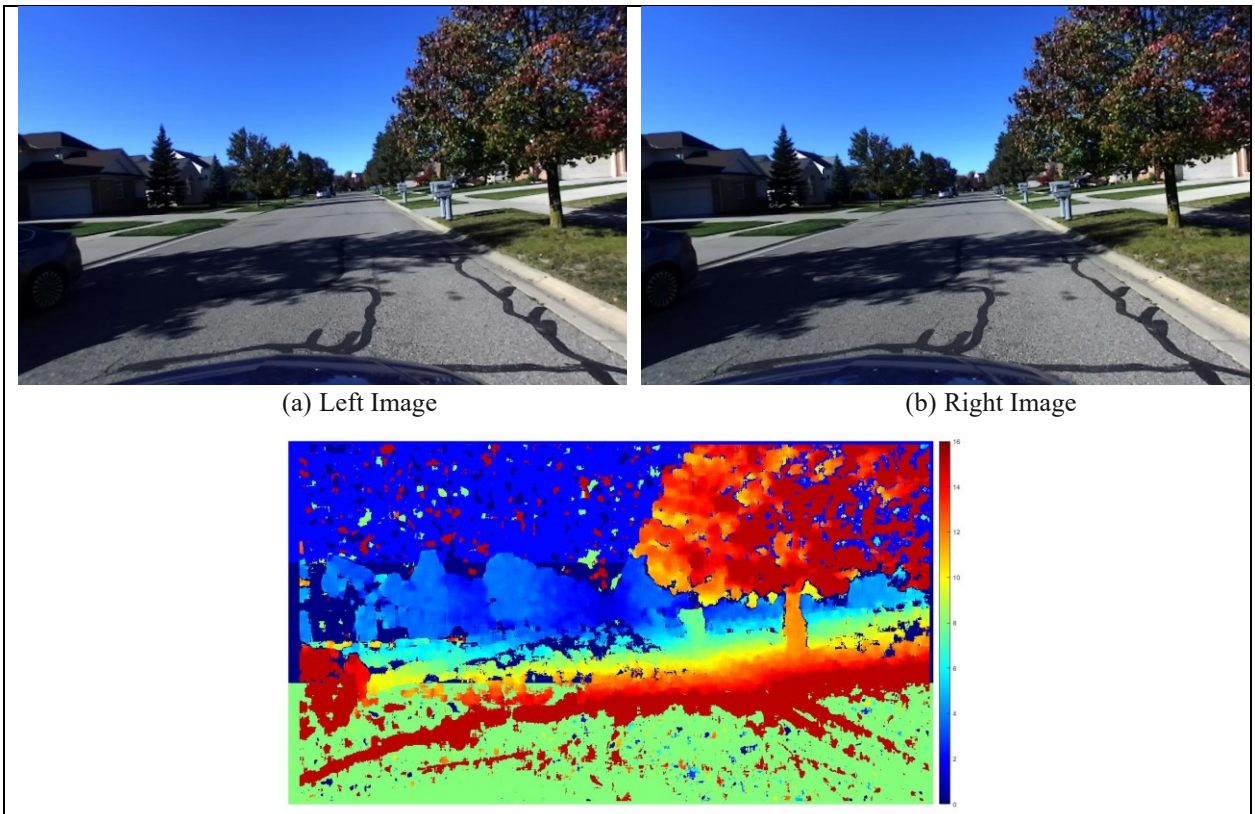
value threshold) in the image plane, were likely to belong to the sky segment, and we assigned to it the lowest disparity value (very far means very low disparity). For segments below a certain level in the image that were missing disparity scores, we assumed they belonged to the road segment, and we gave them a higher disparity value.

C. Calculate disparity maps from each stereo image pair and estimate the distance of objects from the camera.

Objects close to the camera have larger disparity values than objects farther from the cameras. This is because for a light source far from the camera, the light rays are almost parallel to both the left and right camera in the stereo camera system. Light rays emitted from closer objects arrive at the two cameras at different angles, thus are perceived by different areas of the vision sensors in each camera. Mathematically, from equation 4.2,

$$\lim_{z\to\infty} D = \lim_{z\to\infty} \frac{Bf}{Z} = 0 \tag{4.3}$$

Based on this, depth information can roughly be estimated from the disparity map of the image frame. Figure 4-5 shows an example of left and right scene images, along with the calculated disparity map for them using MATLAB disparity function.

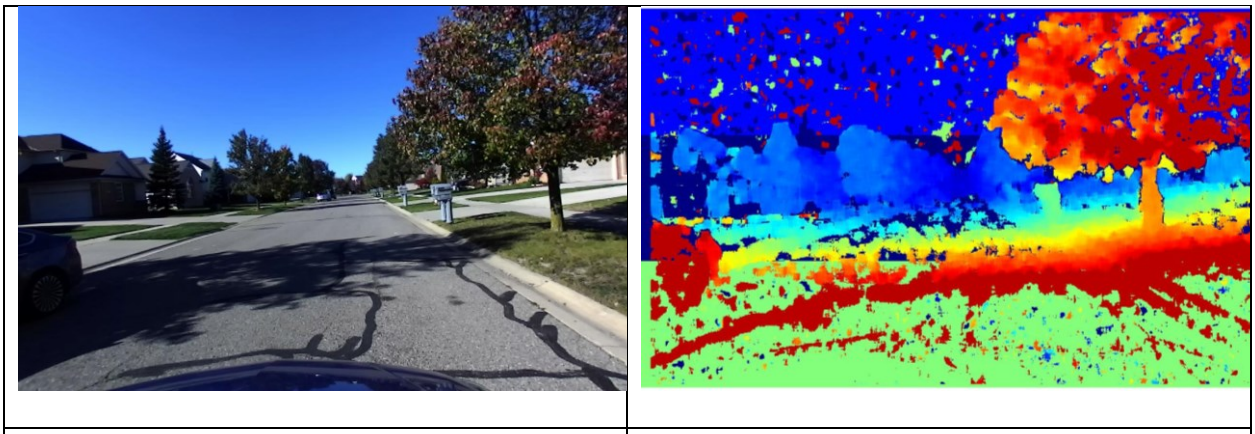


(a) Left Image (b) Right Image

(c) Disparity Map

Figure 4-5: The Disparity map (c) is generated for the left (a) and right (b) image pair

D. Generate random rain streak masks that have the same dimensions as that of the scene images. The model can be configured to generate any number of masks from 2 to16 masks, each representing rain streaks at one distance level from the cameras. Each rain streak starts as a random pixel in a mask buffer. A series of erosion and dilation processes are applied to these random images to generate rain streaks. We varied the length of rain streaks and their density as a function of the relative distance represented by each mask. One rain streak mask is generated at the end of this step, by combining the different masks. Figure 4-6 shows a scene image (left camera), the generated disparity map, and rain streak masks at two levels.



(a) Scene Image (left)

(b) Disparity Map

(c) Mask level 2 (far)

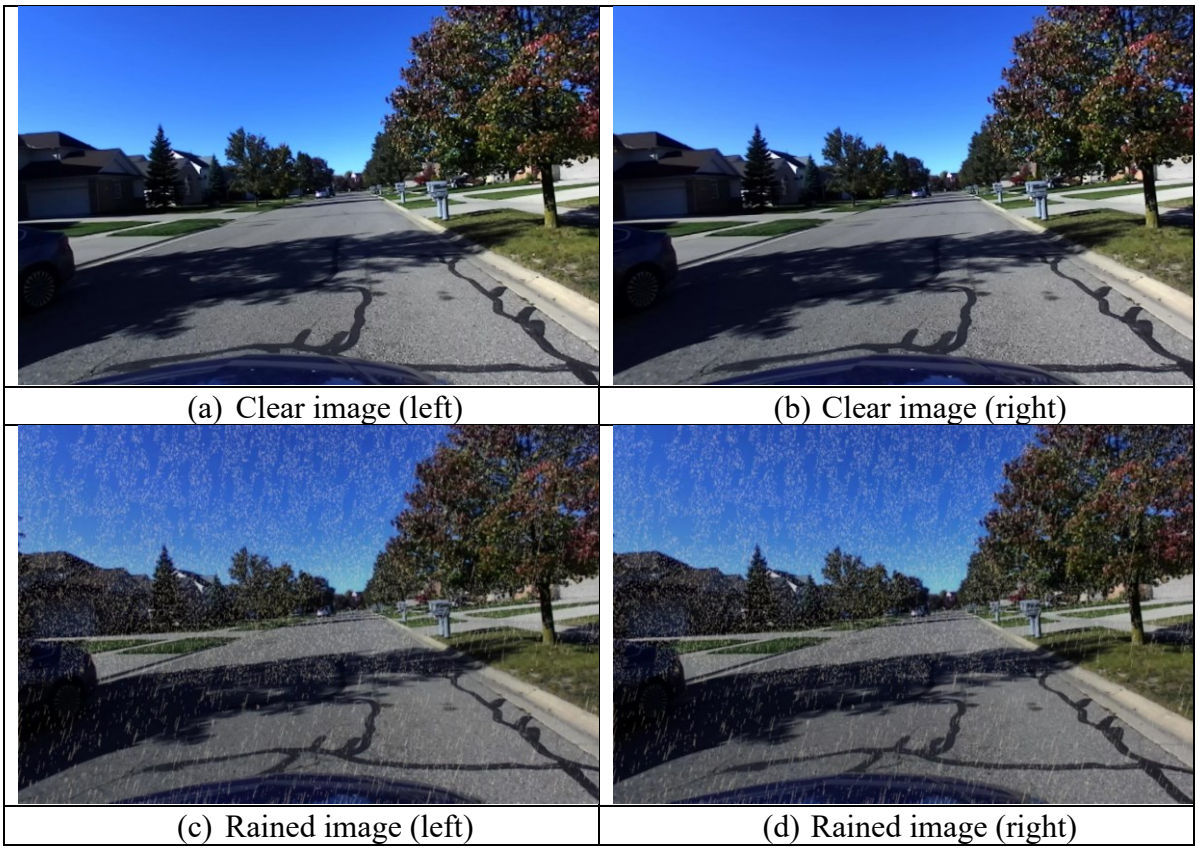
(d) Mask level 8 (near)

Figure 4-6: Based on the disparity map (b) generated for the scene image pair in (a), rain streak masks are randomly generated. Masks representing nearby planes show longer and less dense streaks (d), compared to shorted and more dense streaks for farther planes (c)

E. Add rain masks to the image and apply a local and global blurring effect. In this step, we split the RGB color image into its three image channels (Red, Green, and Blue), then we add the rain mask to each image channel individually. Rain streak brightness is balanced using its surrounding pixels (background). As described earlier, rain streaks tend to be slightly brighter than their background. The blurring effect is then applied to the rain streaks, to account for the distortion caused by the camera lens, as described earlier. The three image channels are finally combined to create one rained image.

F. For images taken under clear weather conditions, we add an overcast effect by reducing the brightness level of the image pixels. This step is not required if the original images were taken under overcast weather conditions.

G. Use the disparity map data to shift the rain streaks generated for the reference image (e.g., left) to the other image. This step is introduced to reduce the time needed for generating masks for left and right images individually. It also guarantees that the rain streaks for the left and right images are matched, and accurately shifted, per the generated disparity map of the clear image pair. Steps E and, optionally F, is then applied to account for photometric characteristics of the rain streaks and rained images. This step (G) is not required if the desired output is a mono rained image rather than stereo image pair. Figure 4-7 shows one image pair before and after rain streaks are added.



(a) Clear image (left) (b) Clear image (right)

(c) Rained image (left) (d) Rained image (right)

Figure 4-7: The falling rain simulator generates rained images (c), (d) from clear images (a), (b). The rain streaks in the left and right rained images are matched according to the disparity map generated for the original clear images.

4.2.2 Experiments and Results

We implemented the falling rain streaks simulator in MATLAB 2018b. for model parameters fine-tuning, we experimented with the following aspects

i. The number of masks. We tested generating a different number of masks, each representing one disparity (inverse depth) level. For our setup, we found that 6 to 8 masks provided a good balance between execution time and sufficiently capturing the distribution of rain streaks in an image, as perceived by a human observer.

ii. Smoothing filter weights. For seamless integration of rain streaks into the original image, the brightness of the rain streak must be adjusted to match, but slightly exceed, those of surrounding pixels. We experimented with averaging the brightness of 8-neighboring and

4-neighboring pixels to the rained image to use it as initial brightness for the streak pixel and found little difference in most cases, so we used the simpler 4-neighboring average. We then adjusted the brightness of the rain streak pixel to be lighter than its surroundings.

iii. Blurring filter weights. Blurring effect is added on local levels around raindrops and also globally to the overall image. We experimented with different filters to implement the desk of confusion idea and found out that a Gaussian filter with sigma = 0.8 was best for local blurring. For global blurring, a Gaussian filter with sigma = 0.3 provided the best visual results.

iv. We experimented with two methods for applying the overcast effect. The first involves bringing the brightness levels closer to the mean value at each color channel, effectively reducing the color content of the image. The other is a simple reduction of brightness by a given factor (e.g., 0.8). The prior showed slightly better results but we ended up using the latter for simplicity.

We compared our generated rained images to those published as part of the KITTI Vision Benchmark Suits [61]. As shown in Figure 4-8, our model generated more visually convincing rain streaks as compared to the KITTI dataset. The original images for the KITTI dataset were captured under overcast conditions which made the overall rained image more realistic.

(a) Example rained image from our Simulator model

(b) Example of a synthetically-generated rained image from KITTI Dataset

Figure 4-8: Our generated rained image versus the KITTI one. The rain streaks in our image are closer to real rain streaks but the KITTI data set was captured under overcast conditions, which made their final images with generated rain visually convincing.

4.3 Rain Detection and Removal Model

4.3.1 Method

We developed our rain detection and removal model based on the following rain streak characteristics

i. Rain streaks can be represented by an elliptical shape with a length-to-breadth aspect ratio (AR) that is bounded by minimum and maximum values.

ii. Rain Streaks roughly have the same orientation that is measured by the angle of the majority axes of the representing ellipse with the horizon.

iii. Rain Streaks pixels are generally brighter than their surrounding background pixels

iv. The brightness of pixels in a rain streak is generally constant from one frame to the other, provided the lack of background movements.

Based on these assumptions, we developed our falling rain detection and removal as follows

A. Identify pixels groups of pixels that are adjacent and are brighter than others in the image. This method is the same one that Garg and Nayar [59] used in their rain streak detection

model, where the difference between the brightness of a rain streak pixels(I_n) at frame *n* and those in the frames before (I_{n-1}) and after (I_{n+1}) can be given as,

$$\Delta I = I_n - I_{n-1} = I_n - I_{n+1} > c \tag{4.4}$$

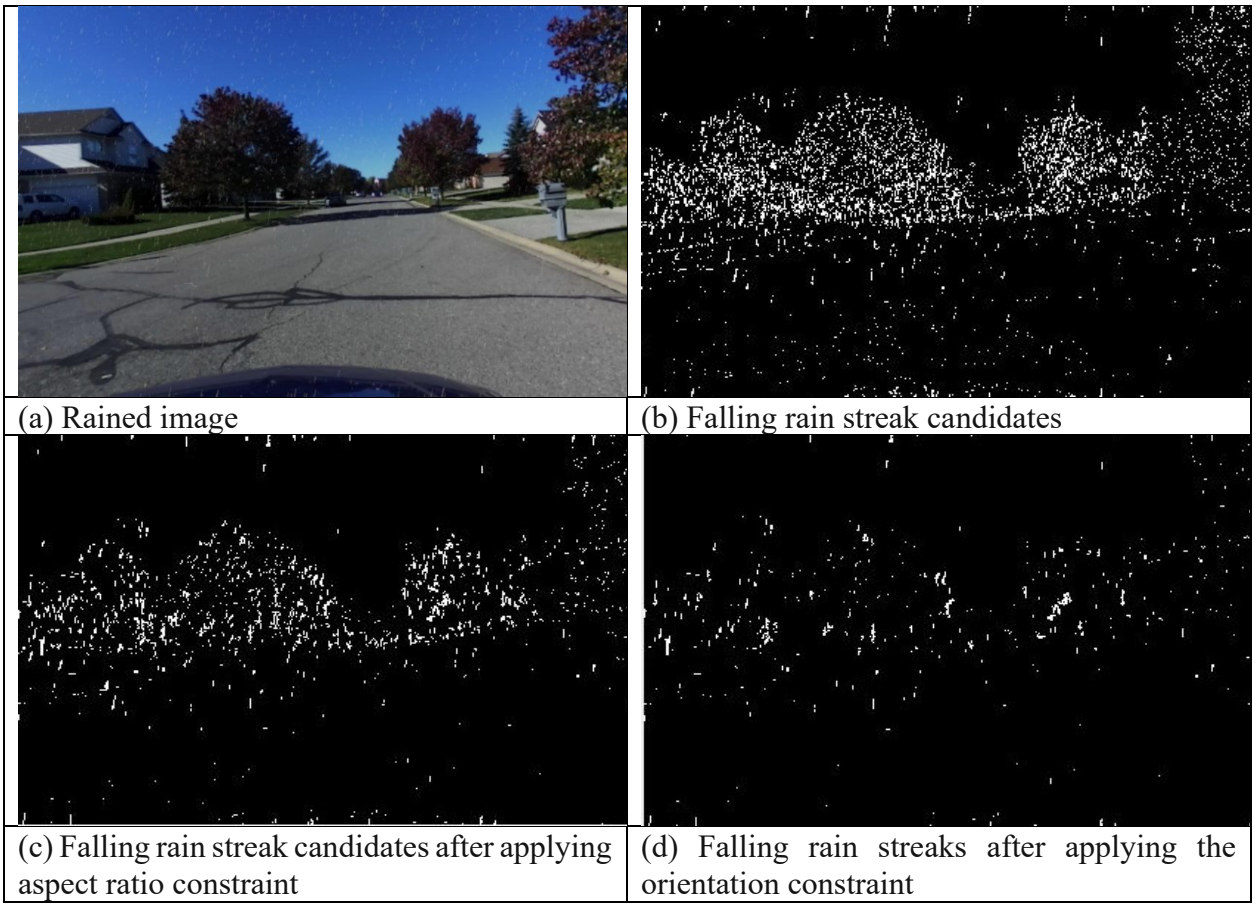
where *c* is a predefined threshold.

We used the "BlobAnalysis" object from the "vision" toolbox in MATLAB 2018B to identify these groups of pixels which are rain streaks candidates. The blob function returns information about these blobs, including their size, majority and minority axes, and orientation.

1) Remove blobs that are too small to be considered raindrops. This step may incorrectly discard very small streaks but these usually do not degrade the quality of the image as much as bigger ones.
2) Eliminate blobs with an aspect ratio that does not fall within the accepted range.
3) Calculate a histogram for the remaining blob orientations and eliminate the outliers that show low occurrences in the histogram
4) The remaining blobs are considered true rain streaks and are eliminated by substituting their pixel brightness by the average of the brightness of the same pixels in the previous and following frames. This is a simple, yet effective, technique for recovering image pixels occluded by rain streaks. The brightness of image pixels from consecutive frames is likely to remain the same or show little change. Averaging the brightness of pixels in the preceding and following frames to the rained frame is thus a reasonable approach. In addition, rain streaks are not likely to linger for more than one image frame so it is logical to assume that the pixels in the preceding and following frames are clear ones. Mathematically, the rain streak pixel substitution process can be given as,

$$I_n = \frac{I_{n-1} + I_{n+1}}{2} \tag{4.5}$$

Figure 4-9 shows the main stages of rain streak detection in our implementation. Potential rain streaks are extracted from rainy images, then aspect ratio and orientation constraints are applied to reject potential "fake" rain pixels.



(a) Rained image | (b) Falling rain streak candidates

(c) Falling rain streak candidates after applying aspect ratio constraint | (d) Falling rain streaks after applying the orientation constraint

Figure 4-9: Rain candidates are generated based on brightness levels only. Aspect ratio constraints and orientation constraints eliminate "fake" rain pixels.

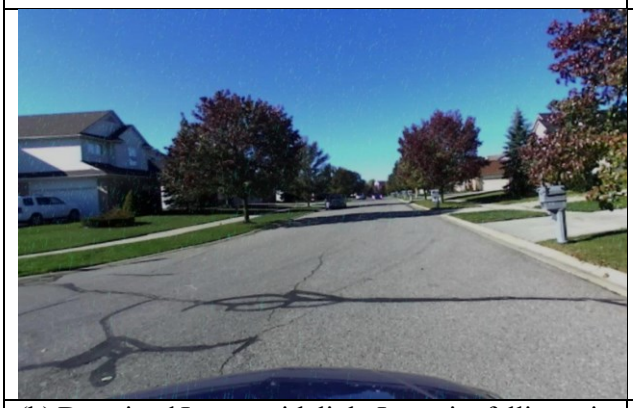
4.3.2 Experiments and Results

To quantify the performance of the rain removal system, we used the PSNR and the SSIM metrics, described in detail in sections 2.2.2 and 2.2.5., respectively.

Figure 4-10 shows images with light and medium intensity levels of simulated falling rain, after applying the de-raining algorithm. Most of the rain in the original image was removed in both rained images. Very little distortion was introduced to the de-rained images, which is an indication that the detection part of the algorithm did not have a lot of false positives. In addition, the images were taken in at a slow-moving ego vehicle, meaning that was little variation in the image background, from one frame to the next. Figure 4-11 shows the SSIM score for a rained and de-rained sequence of 100 images under low and medium rain intensity. We should point out the low and medium classifications of falling rain was done were subjectively selected through visual observation of rain streaks in the images. The average quality improvements from rained to de-rained images, as given by the SSIM were 2.18% percent for medium-intensity rain, and 1.26% for the low-intensity rain dataset. Figure 4-12 shows the PSNR plots for the same 100 samples of rained and de-rained images. unlike the SSIM, the difference in PSNR between rained and de-rained images, both measured against clear images is very small. This is due to the way PSNR is calculated as a pixel-based metric and thus it does not align well with human perception of the image quality.

(a) Rained Image with light Intensity falling rain

(d) Rained Image with medium Intensity falling rain



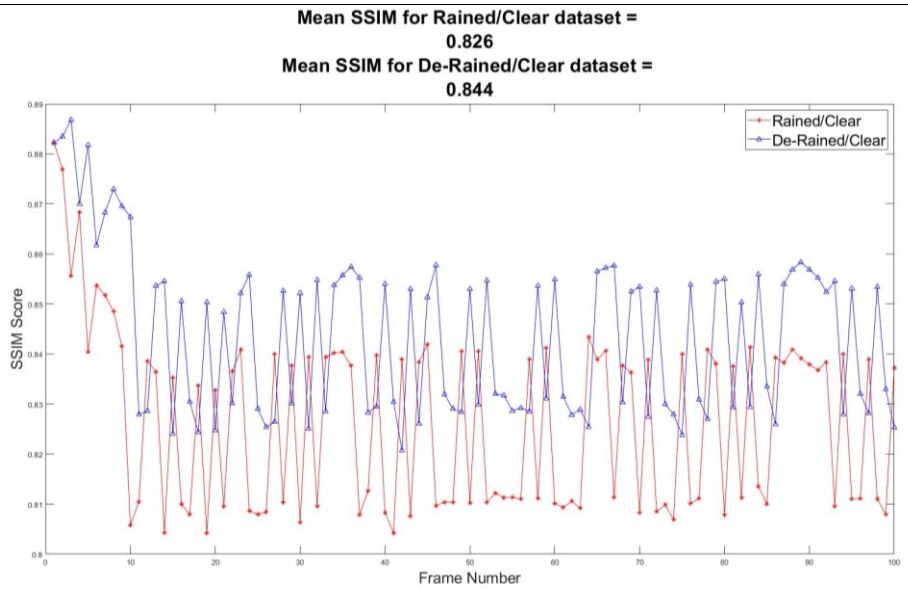
(b) De-rained Image with light Intensity falling rain

(e) De-rained Image with medium Intensity falling rain

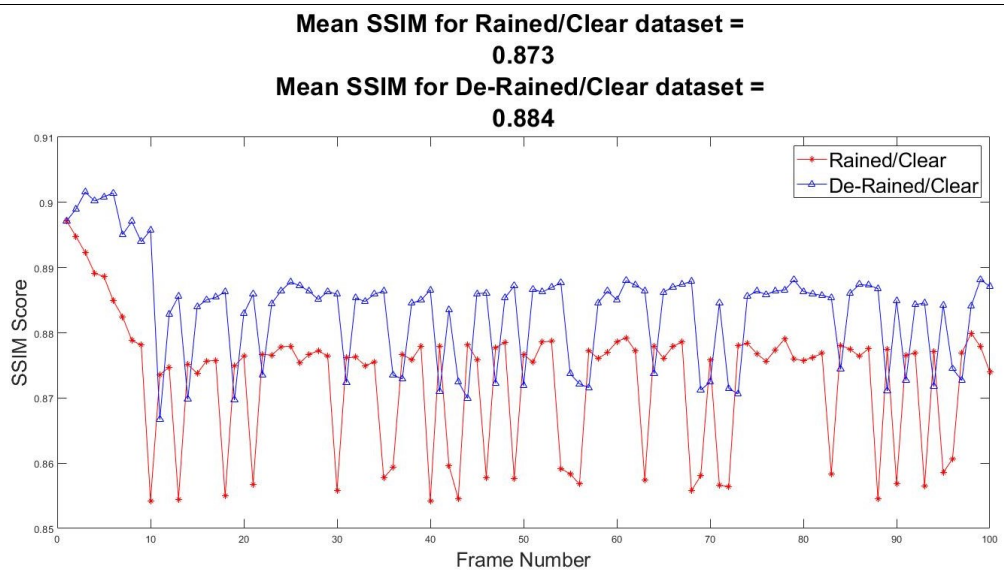
(c) Original rain-free Image

(f) Original rain-free Image

Figure 4-10: Left: Rained, De-rained, and Rain-free Images with light Intensity falling rain (a, b, and c). Right: Rained, De-rained, and Rain-free Images with medium Intensity falling rain (d, e, and f).

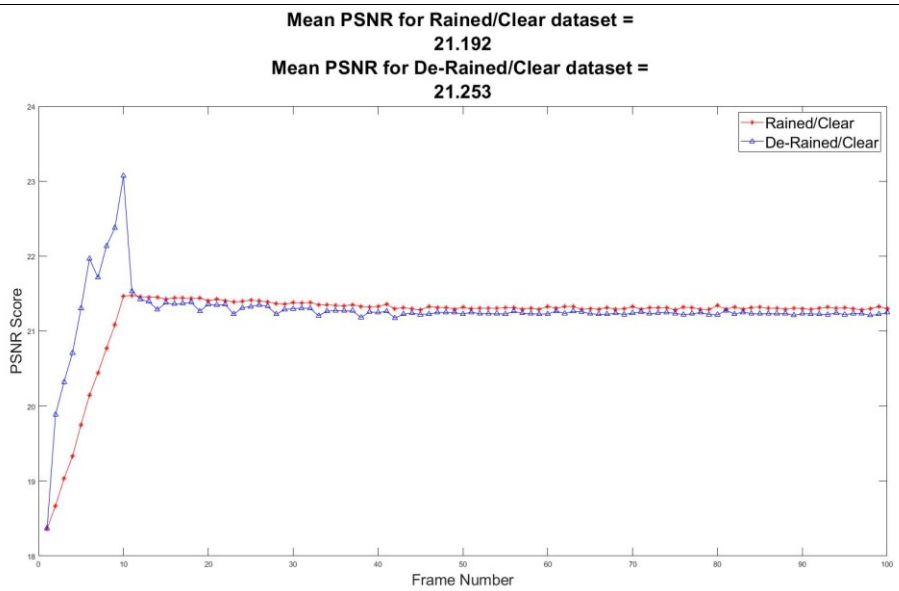


(a) SSIM scores for the medium-intensity falling rain.

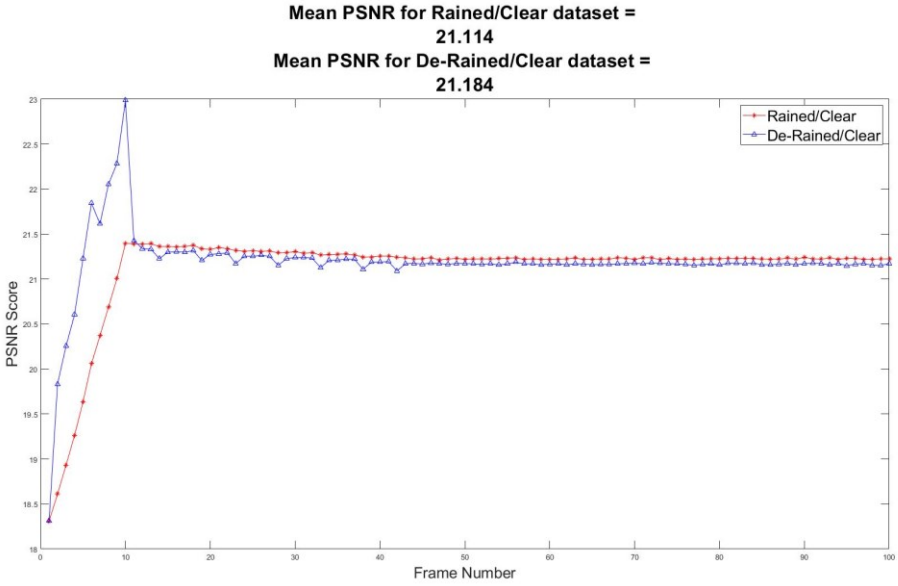


(b) SSIM scores for the low-intensity falling rain.

Figure 4-11: SSIM scores for rained and de-rained images against clear ones, in both low and medium-intensity rain datasets, show that the quality de-rained image in most frames was better than the rained one, identified by a higher SSIM score.



(a) PSNR scores for the medium-intensity falling rain.



(b) PSNR scores for the low-intensity falling rain.

Figure 4-12: PSNR scores for rained and de-rained images against clear ones do not give a conclusive indication of image quality improvement due to removed rain streaks.

4.4 Conclusion

We implemented a framework for simulating and removing rain that can be used to study adverse conditions on the performance of image-based systems. Simulated rain looked realistic both in terms of rain streak size and rain density. There is, however, no technique to quantitatively measure simulated rain vs. real rain. Using disparity maps provided a quick but crude estimation for image depth. This method fell short where disparity values could not be calculated reliably, which is usually, but not always, observed on segments of the image with small variations in intensity and texture. The De-raining system removed most of the visible rain in the images but added distortion. It was clear from test results that de-raining system parameters affect the quality of the de-rained image and the amount of rain removed. The system parameters need to be adjusted according to the density of rain in the image. Utilizing rain density information may help to improve de-raining system performance.

Chapter 5

Effect of Adherent Rain on Vision-Based Object Detection Algorithms

5.1 Introduction

Adverse weather conditions degrade the quality of images used in vision-based advanced driver assistance systems (ADAS) and autonomous driving algorithms. Garg and Nayar [62] broadly classify adverse weather conditions into steady (fog, mist, and haze) or dynamic (rain, snow, and hail). Image degradation takes many forms, depending on the type of adverse condition causing it. Fog and haze, as examples of adverse weather conditions, cause loss of contrast and fidelity in captured images, due to light absorption and scattering in the turbid medium of particles and water droplets in the atmosphere [63, 64].

The degradation effect of haze and fog is due to the aggregate effect of a large number of droplets, despite their small individual droplet size (1 - 10 μm). Raindrops, on the other hand, are larger (0.1-10mm), and their individual effect on image pixels can be visible by the camera [59].

Adherent raindrops onto a vehicle's windshield occlude parts of the input image and blur background texture in regions covered by them. Rain also changes image intensity and disturbs the chromatic properties of color images. Most research work in the field of rain detection and removal focused on the image restoration aspects of the issue, without providing qualitative measures to the effect of input image degradation on the performance of image-based algorithms that use them as their main input.

In this section, we describe the research work we did to quantitatively evaluated the effect of raindrop distortion of input images, on some state-of-the-art deep-learning-based object detection algorithms. We compared each detector's performance with distorted image sets to that using rain-free ones, and provided quantitative scores for performance, using commonly-used quality metrics.

5.2 Method

5.2.1 Data Set

We used ELP Dual Lens Stereo Camera Module (ELP-960P2CAM-V90-VC) to capture MJPEG videos

(2560X960 resolution at 60 frames per second) of real drive cycles. The drive cycles were approximately 40 minutes each and included both highway and local driving segments. A total of 23 videos were captured, 17 of them were during variant levels of rainfall intensities. Figure 5-1 shows a mapped section of these drive cycles.

To create the data sets of image pairs, we captured frames right before and right after that event as the rained (wet) and clear ground truth images, respectively. To reduce the effect of background variations on detection performance, we collected some datasets in a parking lot setup. The background in these datasets was quiz-static, except for some street light fluctuations and the occasional passing of faraway objects. Figure 5-2 and Figure 5-3 show samples of the moving and parked vehicle datasets, under clear and rained conditions.

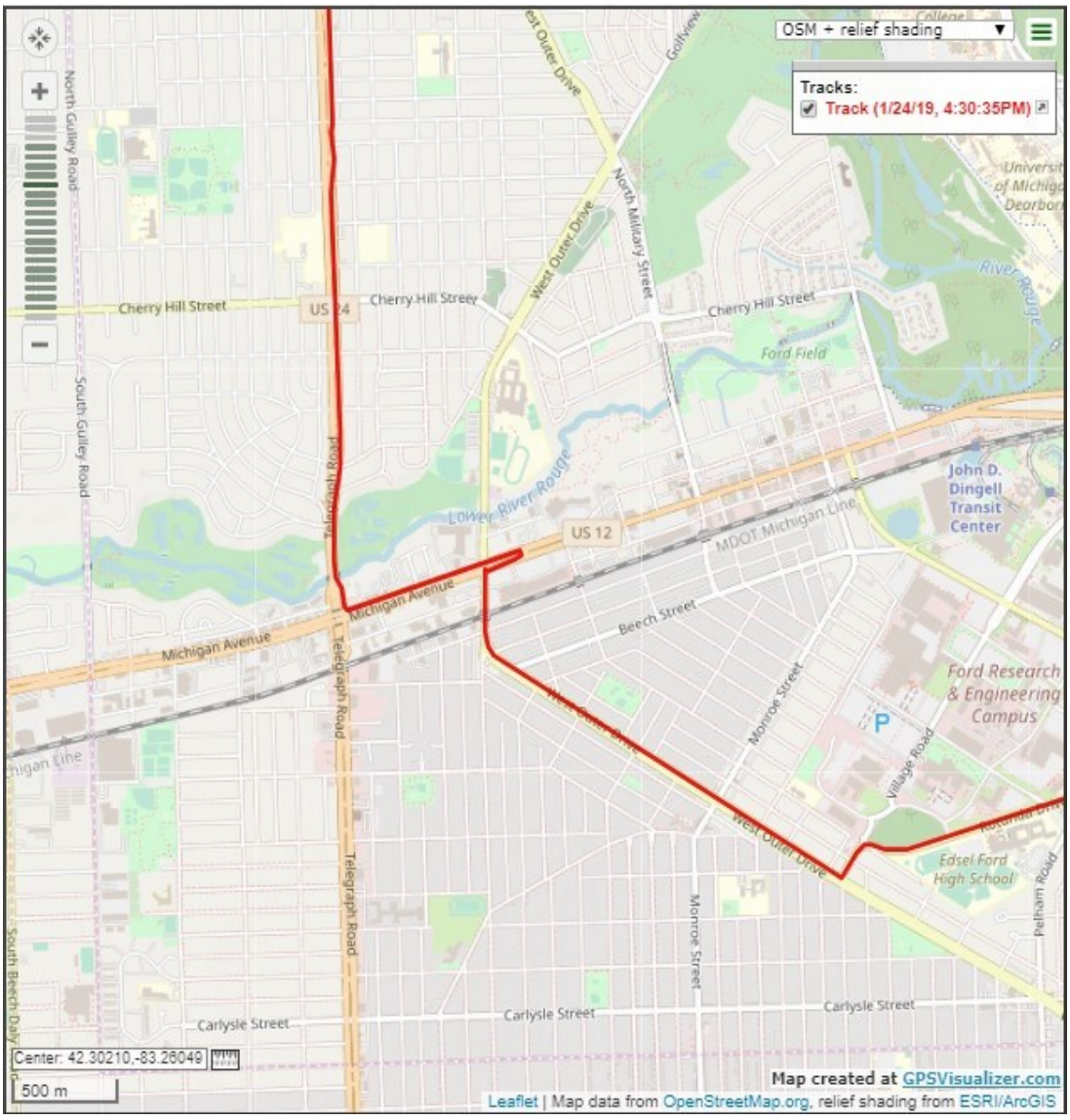


Figure 5-1:Section of the mapped track of test drive cycles.

(a) Rain-free Image

(b) Rained Image

Figure 5-2: Clear and rained image set from the Moving Vehicle dataset

(a) Rain-free Image

(b) Rained Image

Figure 5-3:Clear and rained image set from the Parked Vehicle dataset

5.2.2 Detection Algorithms

We selected three state-of-the-art object detectors to be our test subjects, in evaluating the effect of rain-degraded images on detection performance. The three object detectors we selected Were the Single Shot Detector (SSD) [65], Faster Region-based Convolutional Neural Network (R-CNN) [17], and You Only Look Once Version 3 (YOLOv3) [66].

For the SSD, we used the model from the Wolfram Neural Net Repository implementation [67], which was based on the SSD-VGG-300 architecture, and used a combination of the PASCAL VOC2007

(http://host.robots.ox.ac.uk/pascal/VOC/voc2007/) dataset and PASCAL VOC2012 (http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html) dataset for training.

For the Yolov3 model, we used the Wolfram Neural Net Repository implementation [68], which used the Open Image dataset (https://storage.googleapis.com/openimages/web/index.html) for training.

For the Faster R-CNN detector, we used the implementation by Chen et al. [69], which was based on ResNet50 feature encoder architecture, and trained with the COCO dataset (https://cocodataset.org/#home).

5.2.3 Quality Metrics

To evaluate performance, we used two types of metrics. To assess the image quality and provide a quantitative measure of the image distortion due to rain, we used the Structural Similarity Index Metric (SSIM).

As described by Wang et al. [9], SSIM is an image quality assessment measure designed around the human visual system (HVS). It makes use of structural information change to provide an approximation to perceived image distortion. Unlike error-sensitivity approaches, such as Mean Squared Error (MSE) [70], that estimate image quality degradation using perceived errors, SSIM measures degradation as the level of variations in image structural information [9]. SSIM is a commonly-used metric that was used by many researchers to evaluate image quality (see for example, [11] [47]). SSIM is calculated as a combination of luminance l, contrast c, and structure s comparator functions of two images. The structure similarity index for images x and y can be given by [9],

$$SSIM(x,y) = [l(x,y)]^{\alpha}.[c(x,y)]^{\beta}.[s(x,y)]^{\gamma} \quad (5.1)$$

where $\alpha, \beta,$ and γ are configuration parameters that control the contribution level of each comparator to the overall index (they usually are set to 1). For a more detailed description of the SSIM, please refer to section 2.2.5.

To evaluate the performance of the detector as a function of detected objects in each frame, we used Precision and Recall metrics.

Precision is classified as positive predictive values (PPV) and it represents the portion of positive results that are true positive [71]. It is given in terms of True Positive (TP) and False Positive (FP) predictions as,

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

In our experiments, *TP* represents the number of objects correctly detected in a test frame, and FP is the number of objects incorrectly detected in the same test frame, both relative to objects detected in the ground truth frame.

Precision is a commonly used metric for the assessment of vision-based algorithms (see, for example [30], [35]), but it is usually used in combination with the Recall metric. For a more detailed description of the Precision metric, please refer to section 2.3.1.

Recall is also known as sensitivity and it represents the fraction of relevant instances that have been retrieved over the total amount of relevant instances [71]. It is given by,

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

where FN (False Negative) represents the number of objects not detected in the test frame, as compared to the ground truth frame. For a more detailed description of the Precision metric, please refer to section 2.3.2.

5.3 Experiments and Results

5.3.1 Image Quality Test Results

We created a simple MATLAB script to detect wiper wipe events and used the script to generate test samples. Each test sample is made of one frame before (Wet) and one frame after (clear

ground truth) the wipe event. We used a simple template-matching approach in this wipe event model, where the wiper pixels were used to create different templates at different stages of the wipe event. The algorithm then looked for these templates in each frame of the image sequences and identified the starting and conclusion of the wipe event.

To evaluate the quality of the SSIM as reliable measures of image quality degradation under the rain, we applied this measure on short data sets, each containing one wipe event and few frames before and after that event. The SSIM scores were used as an indicator of the degree of similarity between test images and ground truth images. Figure 5.4 shows the result of measuring SSIM on a moving vehicle series and figure 5.5 shows the result for the idle car series.

The following observations can be made:

i. As the wiping event proceeds, the SSIM score increases the closer we get closer to the ground truth image (rain-free) image frame. This rain-free image is around image frame 202 in the moving vehicle sequence, and around frame 1428 for the idle vehicle series.
ii. As rain starts to accumulate after the conclusion of the wipe event, the SSIM score starts dropping and continues to drop as more rain is accumulated on the windshield.
iii. SSIM scores for moving vehicle series range from 50% for the rained image to 100% for clear one (ground truth). For the parking lot series, the difference was about 20%. The difference in "useful" ranges is attributed to the strong effect of changing background scenes in moving vs. idle vehicles.

Based on the above, we concluded that the SSIM was a reliable metric for assessing the quality of the image and the level of degradation caused by the presence of raindrops. In addition, we focused our analysis on the data sets with a static background (parked vehicle series), to reduce the effect of changing background due to factors other than rain presence in the input images.

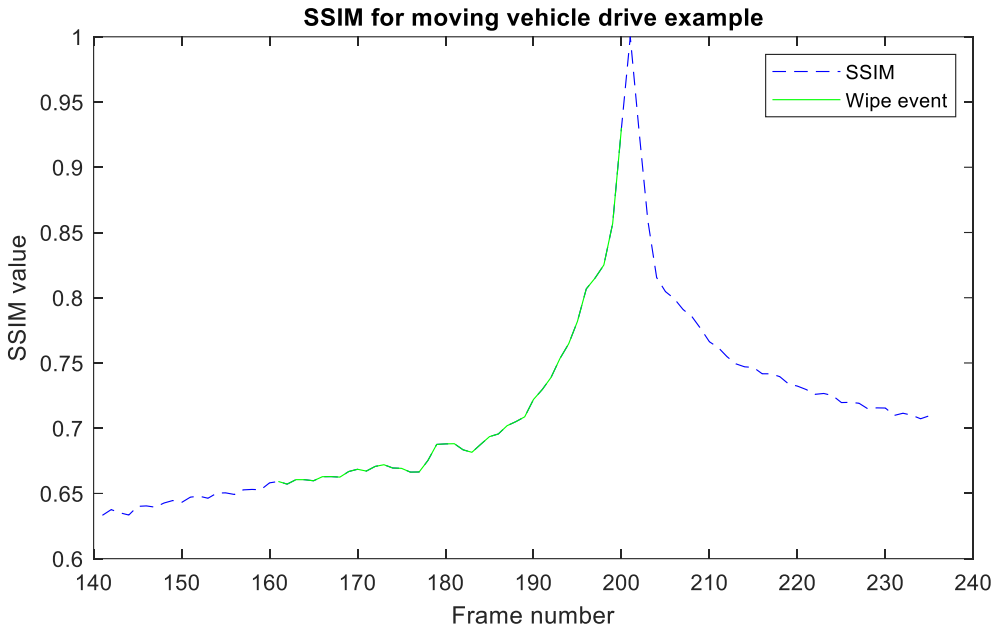


Figure 5-4: SSIM measure for image quality using moving vehicle series, during the windshield wiping event

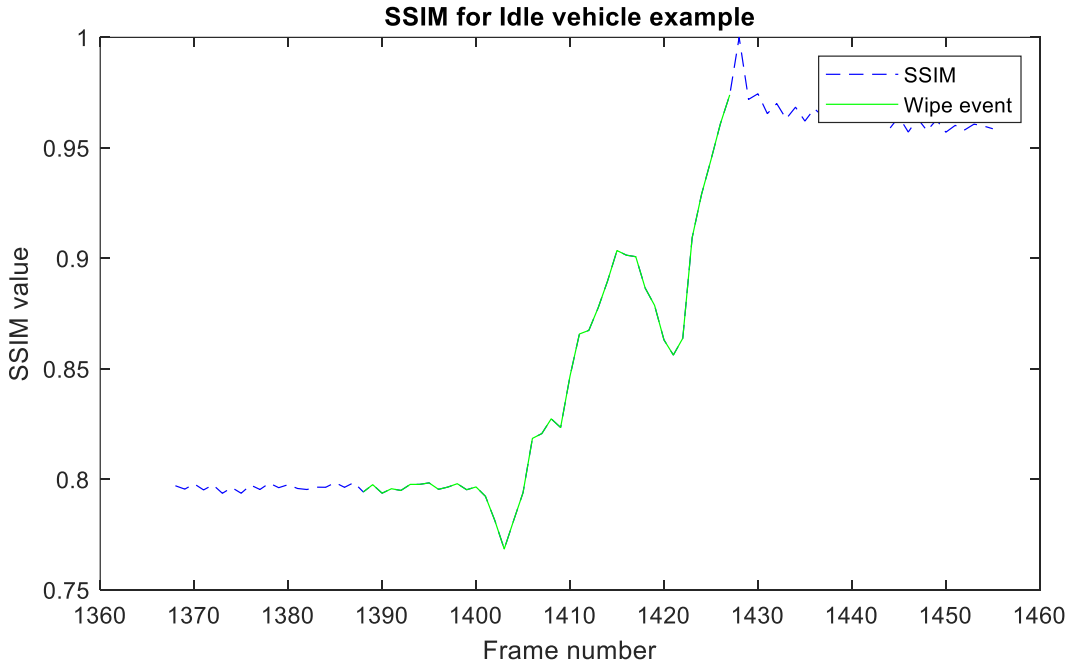
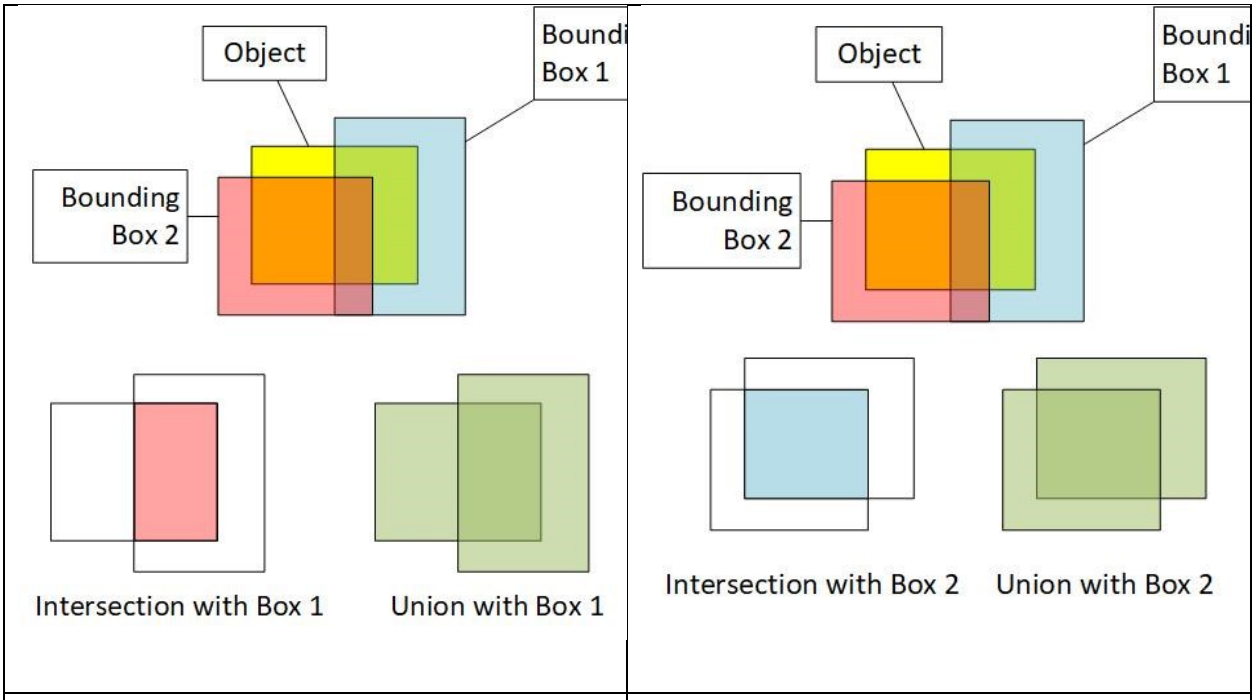


Figure 5-5: SSIM measure for image quality using Parked vehicle series, during the windshield wiping event

5.3.2 Object Detection Test Results

In this section, we present the results of object detection, using different DNN models: SSD [65], Faster R-CNN [72] , and YOLOv3 [66]. We used the parked vehicle datasets in all our

experiments. For all the DNN models used, the parameters Confidence Threshold (CT) and Non-Maximum Suppression (NMS), were set to 0.5 and 0.4, respectively. Only objects that had a detection score greater than or at least equal to the CT were considered in the analysis. The NMS parameter reflects how close the predicted location and size of an object is to its actual location and size in the image. Object detectors use bounding boxes of different sizes and try to match the detected objects to one or more of these boxes. Ideally, there shall be one bounding box per detected object, but it is usually the case that an object can fit inside more than one box. The NMS is used as a metric to optimize the process and select the box that best fits a given object. As shown in Figure 5-6, the NMS value is compared to the Intersection over Union (IoU) score for each Bounding box, which results in selecting Box 1 with a higher IoU than the NMS value, over Box 2.



(a) IoU(Box 1) = 0.293

(b) IoU(Box 2) = 0.238

Figure 5-6: Calculating IoU scores for two different bounding boxes to the same detected object. If NMS = 0.25, then Box 1 would be used rather than Box 2

The results of the three object detectors are shown below.

5.3.2.1 SSD Results

We used the SSD model to detect objects in our captured datasets, for an idle and moving vehicle, and different adherent raindrop densities. Figure 5-7 shows an example of wet and clear images, with objects detected using SSD. We notice that under rainy conditions, the SSD detectors failed to detect many objects in the parked vehicle series and did not detect the truck in the moving vehicle series, compared to the clear image. Also, in the rained moving vehicle image, the SSD incorrectly detected (FN) a boat object in the scene, where lots of raindrop content was observed. Figure 5-8 shows a plot of recall versus SSIM for one parking lot series (a), and the number of detected objects in each image frame, using the SSD object detector. In Figure 5-8 (a), the SSD detector's performance seems to improve as the SSIM increases but saturates around an SSIM score of about 0.82. Using SSIM as an indicator of image quality under different rain conditions, we can conclude that the SSD detector shows robustness to rain-induced image degradation up to a certain level. Looking at this from a different perspective, if de-raining is used to clear input images before they are used by the SSD object detector, then it is sufficient to clean the images to meet the sensitivity level of the detector. Any cleaning beyond this level will not cause a noticeable improvement in the detector's performance.

(a) Example of SSD output on a rain-free image from the Parked vehicle sequence

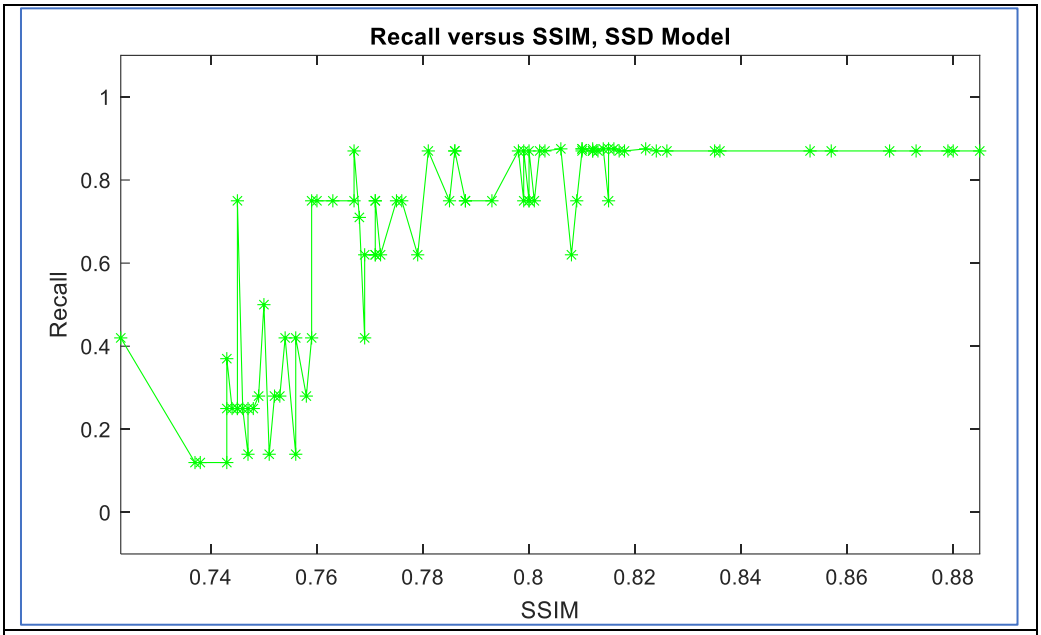
(b) Example of SSD output on a rained image from the Parked vehicle sequence

(c) Example of SSD output on a rain-free image from the moving vehicle sequence

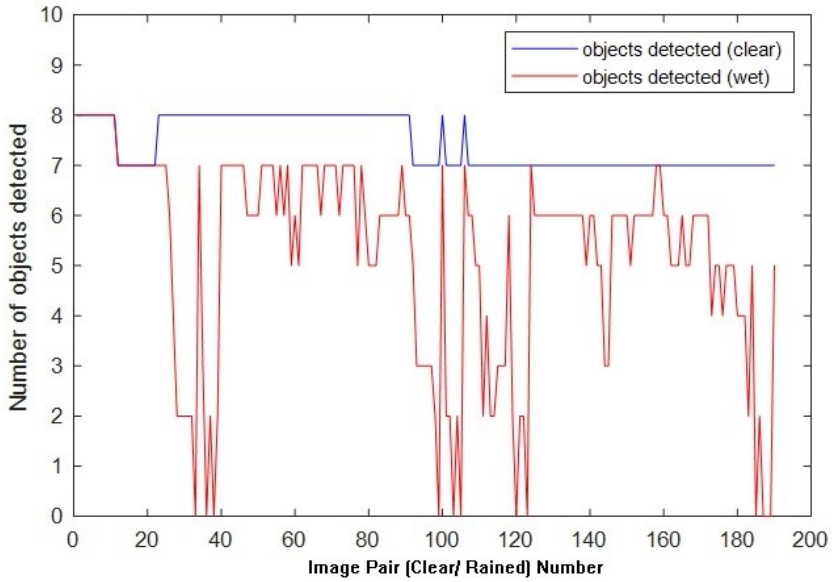
(d) Example of SSD output on a rained image from the moving vehicle sequence

Figure 5-7: More objects are detected using SSD in the clear image (a and c) than in the rained image (b and d). Moreover, some misclassifications are found in the rained image (d). For detected objects, detection confidence in the rained images is lower than that in the clear images.

Figure 5-8 (b) shows that in the clear dataset, the SSD steadily detected seven or eight objects per frame (out of 11 actual objects in the parking lot scene). For the rained dataset, however, the number of objects detected sharply fluctuated between zero and eight, depending on the degradation level (due to raindrop presence) in each frame.



(a) Recall vs. SSIM for SSD object detector, using parked vehicle dataset



(b) Number of objects detected in each image frame by the SSD detector, using one parked vehicle dataset

Figure 5-8: Applying SSD on parking lot series, we observe a clear trend of increasing Recall values with increased SSIM Score (a). In addition, the number of detected objects is almost constant from one frame to the other in the clear image dataset but fluctuates a lot in the rained image dataset (b).

We calculated the correlation between image quality, given as an SSIM score, and the Recall, for different data sets. The results are shown in Table 5-1. For the parking lot series, results show a strong correlation between image quality and object detection performance. For moving car sets,

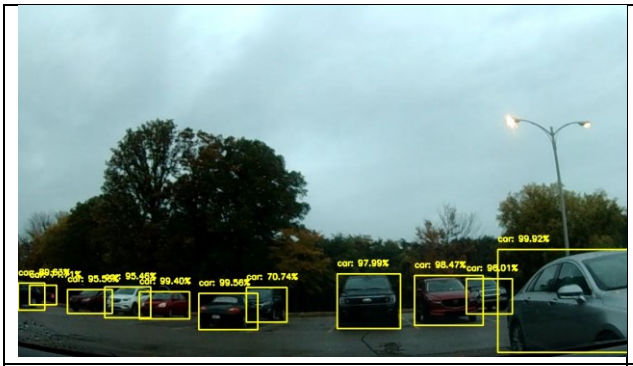
the relation is observable for short data series but is weak in the longer ones. This is because, in the moving vehicle case, the changing background plays a bigger role than the raindrop presence deviating a given image frame from the series reference frames. The longer the series is, the move variations in the background occur and the less similar a frame becomes to the reference frame. Since the SSD detector's performance is still dependent on the input image quality, the correlation between the recall and the SSIM scores is low for moving vehicle series.

Table 5-1: The correlation coefficient between image quality and detection performance for different data series, using the SSD object detector.

Data Series ID	Vehicle condition	Correlation coefficient	Number of Images
16_52_17_Pro_L	Idle	0.7560	84
16_52_17_Pro_LR	Idle	0.7600	168
16_21_25_Pro	Moving	0.5596	117
16_19_56_Pro_R	Moving	0.0580	310

5.3.2.2 Faster R-CNN Results

Using the Faster R-CNN detector, we repeated the test we had conducted using the SSD detector, using both parked and moving datasets. Figure 5-9 shows an example of wet and clear images, with objects detected using Faster R-CNN. We notice that under rainy conditions, the Faster R-CNN detectors failed to detect many objects in both the parked and moving vehicle series. Figure 5-10 shows a plot of recall versus SSIM for one parking lot series (a), and the number of detected objects in each image frame, using the Faster R-CNN object detector (b). As in the SSD case, we observe a trend of increasing recall as the image quality (given as SSIM score) increases, and that the recall scores saturate after some SSIM value (around .835), and any increase in SSIM values after that does not translate to noticeable recall score.



(a) Example of Faster R-CNN output on a rain-free image from the Parked vehicle sequence

(b) Example of Faster R-CNN output on a rained image from the Parked vehicle sequence

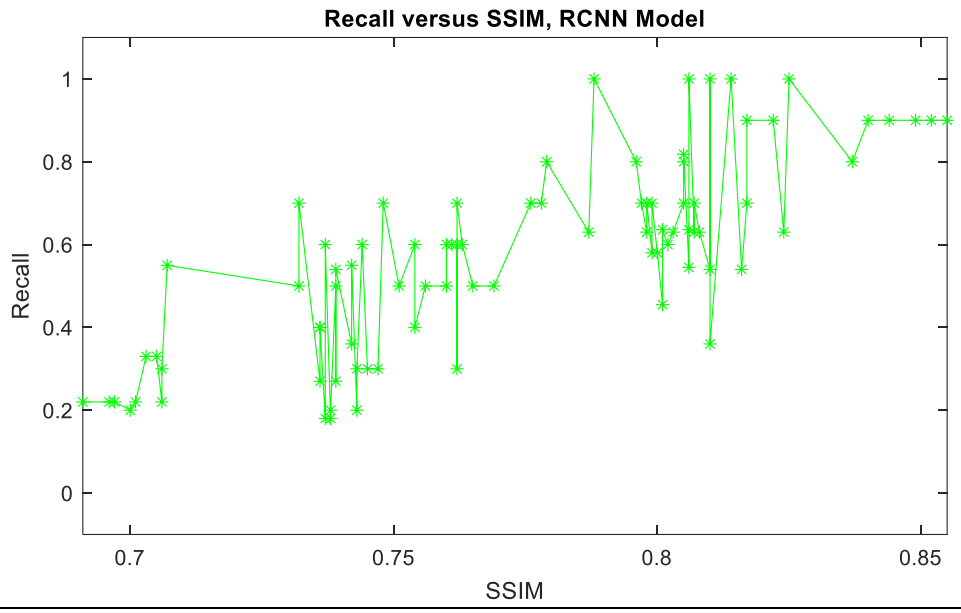
(c) Example of Faster R-CNN output on a rain-free image from the moving vehicle sequence

(d) Example of Faster R-CNN output on a rained image from the moving vehicle sequence

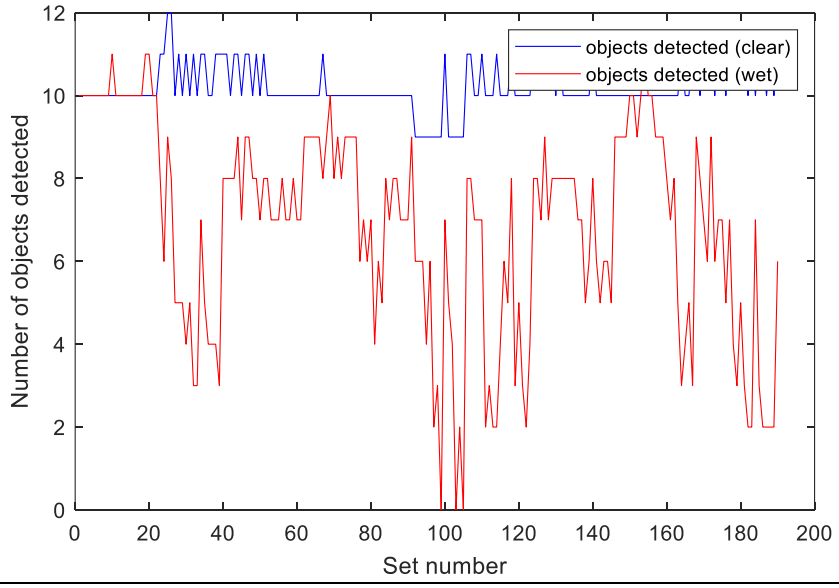
Figure 5-9: More objects are detected using Faster R-CNN in the clear image (a and c) than in the rained image (b and d). For detected objects, detection confidence in the rained images is lower than that in clear ones.

We also observe that there are more fluctuations in the recall scope versus SSIM score than what is observed in the SSD case, using the same data sets. This might be an indication that the Faster R-CNN model we used is more susceptible to image quality degradations than the SSD model.

As for the number of detected objects per frame, Figure 5-10 shows that the Farter R-CNN detector performs slightly better than the SSD one on the same datasets tested, with detected objects ranging from nine to twelve per frame. The detected objects in the rained dataset show lots of fluctuations from one frame to the other, based on the amount of rain (image degradation level) in each frame.



(a) Recall vs. SSIM for Farter R-CNN object detector, using parked vehicle dataset



(b) Number of objects detected in each image frame by the Farter R-CNN detector, using one parked vehicle dataset

Figure 5-10: Applying Farter R-CNN on parking lot series, we observe a clear trend of increasing Recall values with increased SSIM Score (a). In addition, the number of detected objects is almost constant from one frame to the other in the clear image dataset but fluctuates a lot in the rained image dataset (b).

As before, we calculated the correlation between image quality, given as an SSIM score, and the Recall, for different data sets. The results are shown in Table 5-2.

Table 5-2: The correlation coefficient between image quality and detection performance for different data series, using the Faster R-CNN object detector.

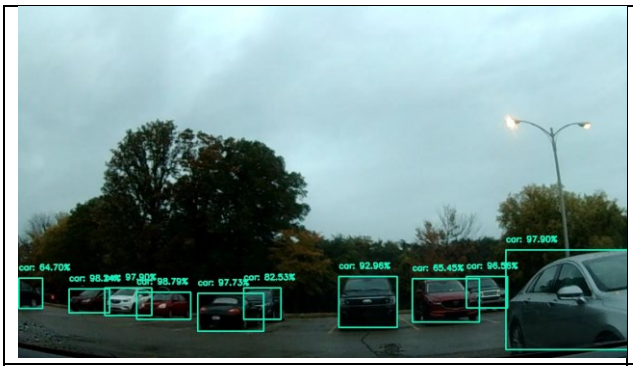
Data Series ID	Vehicle condition	Correlation coefficient	Number of Images
16_52_17_Pro_L	Idle	0.6857	84
16_52_17_Pro_LR	Idle	0.7157	168
16_21_25_Pro	Moving	0.6091	117
16_19_56_Pro_R	Moving	0.0495	310

For the parking lot series, results show a strong correlation between image quality and object detection performance. For moving car sets, the relation is still observable for short data series but is weak in the longer one, similar to the relations we got using the SSD detector. In general, the Faster R-CNN detected more objects on average in both rained and clear datasets, as compared to the SSD and YOLOv3 models we used.

5.3.2.3 YOLOv3 Results

We repeated the same experiments as before using the YOLOv3 object detector. Just like in the case of SSD and Faster R-CNN, Figure 5-11 shows the YOLOv3 detecting a smaller number of images in the wet samples than the clear ones. The detection confidence levels for images in the wet images are lower than those in clear ones. Figure 5-12 shows similar trends to those observed with the SSD and Faster R-CNN object detectors.

Table 5-3 also shows that, as in the SSD and Faster R-CNN, the correlation between detection performance and image quality is strong for the parked vehicle datasets but weaker for moving vehicles using long image sequence as an input.



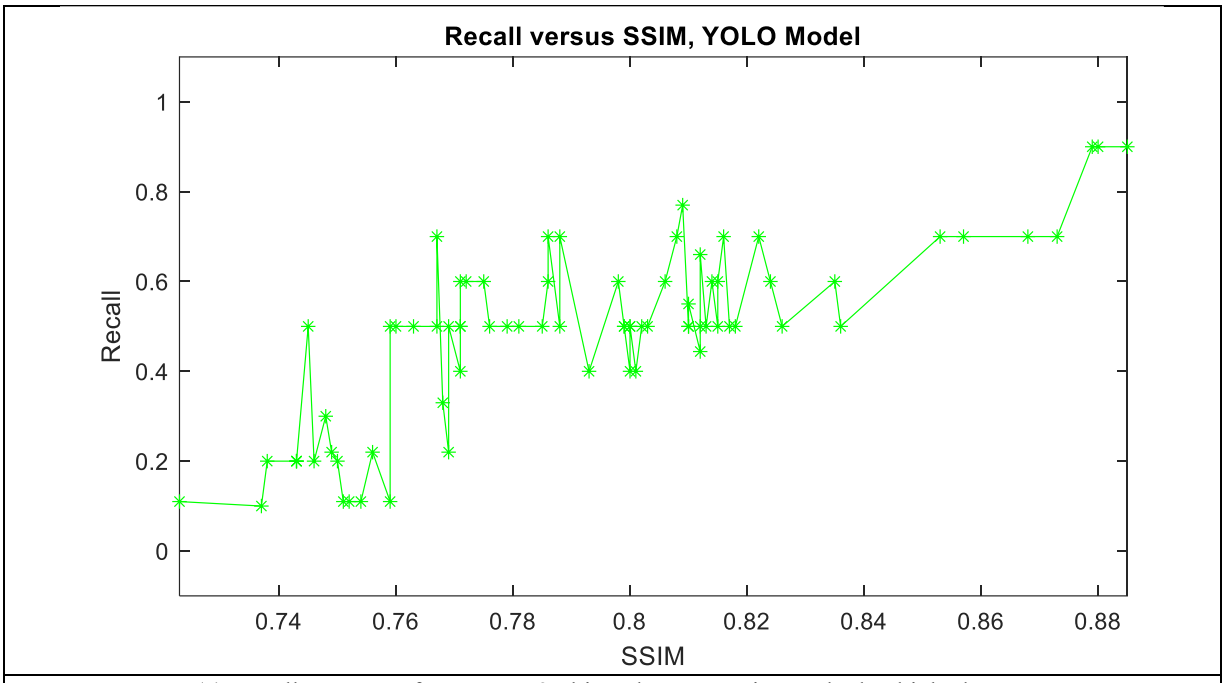
(a) Example of Faster YOLOv3 output on a rain-free image from the Parked vehicle sequence

(b) Example of Faster YOLOv3 output on a rained image from the Parked vehicle sequence

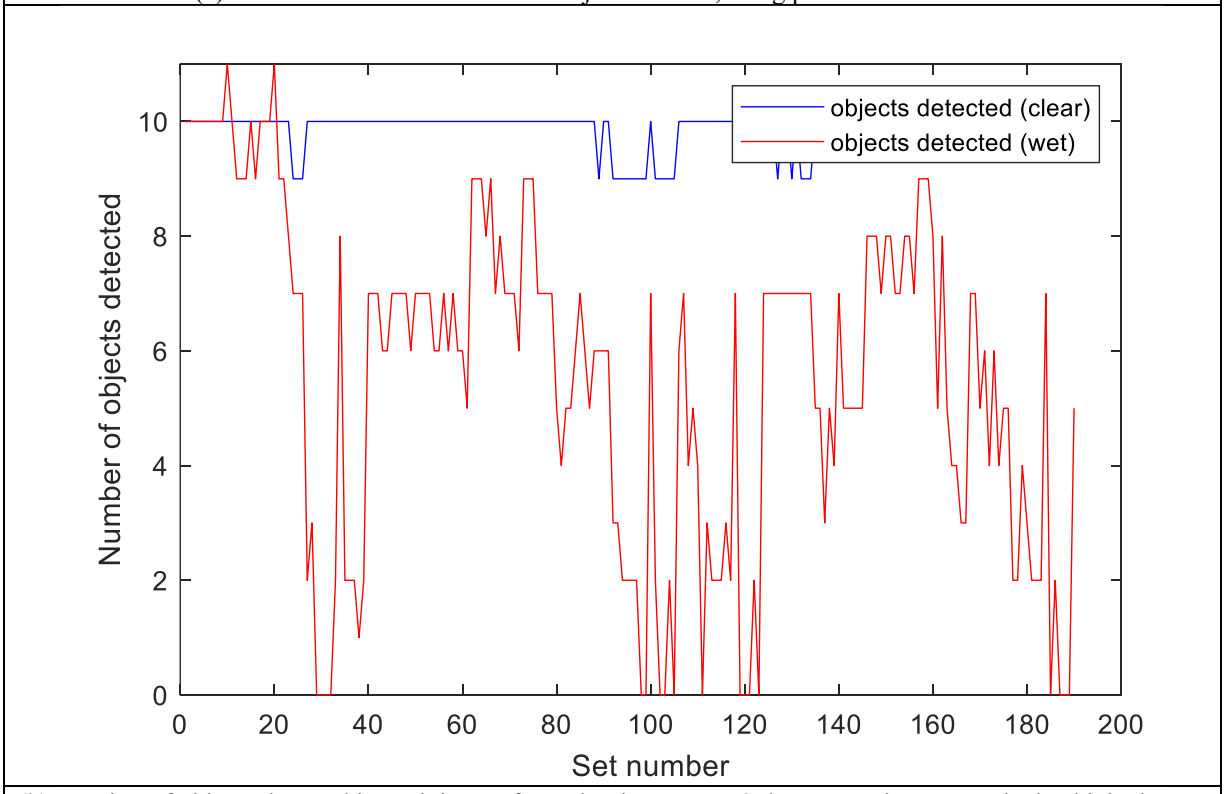
(c) Example of YOLOv3 output on a rain-free image from the moving vehicle sequence

(d) Example of Faster YOLOv3output on a rained image from the moving vehicle sequence

Figure 5-11: More objects are detected using YOLOv3 in the clear image (a and c) than in the rained image (b and d). For detected objects, detection confidence in the rained images is lower than that in clear ones.



(a) Recall vs. SSIM for YOLOv3 object detector, using parked vehicle dataset



(b) Number of objects detected in each image frame by the YOLOv3 detector, using one parked vehicle dataset

Figure 5-12: Applying YOLOv3 on the parking lot series, we observe a clear trend of increasing Recall values with increased SSIM Score (a). In addition, the number of detected objects is almost constant from one frame to the other in the clear image.

Table 5-3: The correlation coefficient between image quality and detection performance for different data series, using the YOLOv3 object detector.

Data Series ID	Vehicle condition	Correlation coefficient	Number of Images
16_52_17_Pro_L	Idle	0.7793	84
16_52_17_Pro_LR	Idle	0.8750	168
16_21_25_Pro	Moving	0.4087	117
16_19_56_Pro_R	Moving	0.0099	310

5.4 Analysis and Discussion

As shown in the previous section, there is a strong relation between image quality, presented in the form of SSIM value, and the performance of object detectors. For the parked vehicle series, results show a rather strong correlation between image quality and object detection performance. For moving car sets, the relation is observable for short data series but is very weak when considering long series. As discussed earlier, variations in objects content in wet and clear images negatively affect both image quality (similarity) calculations and object detection performance. The longer the series is, the less reliable the correlation coefficient gets as a means of establishing a relationship. In terms of object detectors, we observed that, in general, more objects were detected by YOLOv3 and Faster R-CNN than with SSD. This can be partially attributed to the larger class of objects these two detectors are trained to detect (COCO labels), as compared to SSD (PASCAL VOC labels). One more thing to consider is that since we are using the three object detectors as "test instruments" to study the relation between raindrop-degraded input images and the performance of vision-based systems, any deficiency in the design of the test instrument is in effect a latent variable. Each of the detector models we used seemed, to suffer some inconsistencies in the detection results. In some data series, for example, the number of objects detected in two consecutive frames was noticeably different, even though the scene

setup, raindrops, and background objects, did not change much between the two frames. In other cases, a detector would identify an object in the rained images but fail to detect the same object in the matching clear image frame. We predict that this latent variable has an effect on the correlation measure but there is no easy way to measure that effect. No clear pattern or bias was observed for this latent variable representing imperfections in the detection accuracy. To reduce the effect of this latent variable we calculated the correlation scores based on a large size of image samples.

5.5 Conclusion

We studied the effect of image degradation due to the presence of adherent raindrops on the performance of a state-of-the-art object detection algorithm. We used SSIM and Recall metrics to assess the quality of rained images relative to the clear ones for the same image scenes. In the absence of moving objects in the background of the image sequence datasets, both metrics showed a strong correlation between the image quality and the presence of adherent raindrops. We then used the clear and rained image datasets as inputs to the object detention models and recorded the number of detected objects per frame and the detection confidence for each object. We also calculated the Recall score and used it as an indicator of the object detector performance. We then analyzed the relationship between image quality (SSIM and Recall) and the detector performance (Recall, number of objects, and detection confidence level). A strong correlation score was calculated between the image quality and detector performance for all the object detectors we used in our experiments. One interesting observation was that all the object detector models we tested had a level of tolerance to the image degradation due to rain. This level was different from one detector to the other. The difference is likely due to the design of the individual detectors, as well as the implementation of the designs that we used in our work. If a de-raining stage was used

before the object detection system, it is likely, based on this observation, that the detectors would yield satisfactory results (for the target system), even with less-than-perfect de-raining results. As a result, smaller models and less training time for the de-raining models could be used, without sacrificing the performance of the target vision-based system, being the object detectors in our case.

Chapter 6

Dynamic Adherent Raindrop Simulator for Automotive Vision Systems

6.1 Introduction

Most vision-based systems developed for automotive applications assume optimal visibility conditions. Deviations from these optimal conditions usually result in performance degradations or complete failure of vision-based systems. Reduced lighting level, for example, causes performance degradations in intensity-based vision algorithms, and may cause a total system failure in color-based algorithms that are usually more susceptible to illumination level variations. Raindrops that adhere to the vehicle windshield blocks certain zones of the image and introduce lens effects that cause both spatial and dynamic distortions to the image.

The automotive environment is unpredictable in general. Testing vision-based automotive systems, to verify their robustness against noise factors requires collecting a great deal of data, to cover all possible operational conditions. Collecting representative rained image data is not optimal, since both raindrop sample properties and scene background are uncontrollable. It is not possible to control the size and intensity of real adherent raindrops. This means that many datasets of rained images need to be collected, analyzed, and classified based on adherent raindrop characteristics, before being used for robustness testing and system optimization. Lack of background controllability means that the clear-image ground-truth cannot be established, since it is not possible to repeat the exact drive cycle with and without rain, due to variations in background elements in different drive cycles. De-raining of rainy images presents an option for

estimating ground-truth, rain-free, data. This, however, is not an optimal solution either. De-raining algorithms cannot remove all existing raindrops in an image with high accuracy and reliability. They also add distortion, in terms of incorrectly de-raining clear sections of a rained image and adding spatial and intensity distortions to the de-rained image. In this section, we present a rain simulator system, that adds rain to clear images, collected from real drive cycles. The system is dynamic, meaning that it shows the progressive accumulation of adherent raindrops on a vehicle windshield. The amount of rain and rate of accumulation is controllable, to provide the most flexibility for generating test sets at different rain conditions. Moreover, this section expands on and follows some of the approaches used in our prior work [24] to assess the effects of adherent Rain on deep learning-based object detectors, and compares it with simulated dynamic adherent rain.

6.2 Method

6.2.1 Data Collection

For data collection, we used a dual-lens stereo camera (ELP-960P2CAM-V90-VC) that was attached to the vehicle dashboard, approximately 10 cm away from the windshield. We captured around 15 hours of videos of real drive cycles, under clear and rainy conditions, at 60 frames/second rate and 1280X960 resolution per image frame. We wrote an algorithm in MATLAB scripting language to detect the beginning and end of the wipe events. The frames previous to a wipe event were captured as rained image samples, and the few frames right after that event were considered to represent the clear reference images. Figure 6-1 shows an example of clear (a) and wet (b) images from the data sets.

6.2.2 Quality Metrics

We used two similarity metrics to test the closeness of images with real vs. simulated rain, namely the Structural SIMilarity (SSIM) index, and the Earth Mover Distance (EMD). A description of the SSIM can be found in section 2.2.5.

We used the "ssim" function as implemented in MATLAB 2018-b.

(a) Clear Image | (b) Rained Image

Figure 6-1: Example of captured image sets, clear(a) and wet (b).

The EMD "is a measure of distance between two probability distributions over a region D" [15]. A description of the EMD can be found in section 2.2.6.

We also developed a MATLAB script to calculate Precision and Recall measures for object detector performance with real and simulated rain input. A description of the Precision and Recall metrics can be found in sections 2.3.1 and 2.3.2, respectively.

6.3 Adherent Rain Simulator

We start with the following assumptions while designing our adherent raindrop simulator:

1) Adherent raindrops can take many irregular shapes, but they can be approximated with an ellipse, as a starting point.

2) An Adherent raindrop acts as a lens, adding fish-eye or barrel distortion to the image

3) Adherent raindrops in an image are blurry and lack clear borders that define their shapes.

4) Adherent raindrops are semi-static, in the sense that there is a very little observed movement of a raindrop from one frame to the next.

Figure 6-2 shows the main stages of our raindrop generation process, which are described as follows:

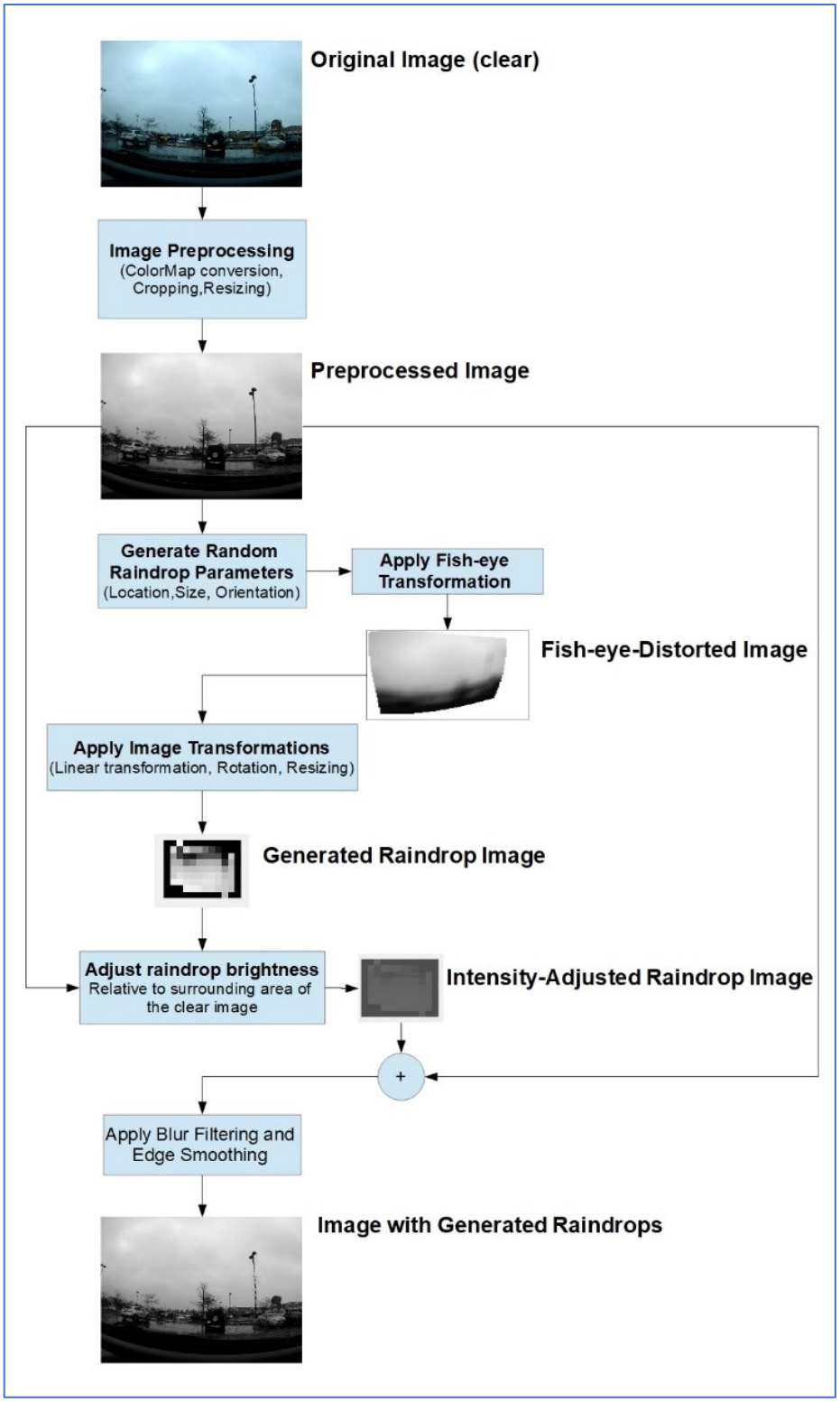


Figure 6-2: Main stages of raindrop generation include image preprocessing, barrel (fisheye) transformation, raindrop image processing, brightness adjustment, and blurring and edge smoothing.

6.3.1 Select Raindrop Shape, Size, and Position

As shown in Figure 6-3, adherent raindrops can come in different shapes and sizes, and align in any possible orientation. We start with an ellipse to approximate the shape of the adherent raindrop (Figure 6-4). Subsequent steps distort this ideal elliptical shape, adding more realism to the simulated raindrop shape.

The size, orientation, and position of raindrops in each frame are arbitrarily selected from a calibratable raindrop characteristics table. Table 6-1 shows the raindrop calibration parameters, with some example ranges.

Figure 6-3: Adherent raindrops can come in different shapes, sizes, and orientations. Photo by Good Stock Photos.

(a) Initial clear image (b) Simulated Raindrop position

Figure 6-4: Starting with a clear image frame (a), the simulator generates arbitrary values for simulated raindrop location, size, and orientation (b).

Table 6-1: Calibration parameters for generating simulated raindrops for each image frame

Parameter	Description
DropsPerFrame	Number of raindrops added to a single frame [1-3]
DropPosition	Position of a raindrop (default is the whole image area)
DropRotation	The orientation of a raindrop [80° - 150°]
DropSize	Size of a raindrop defined in terms of major and minor axes of an ellipse in pixels [10-35 x 3-10]

6.3.2 Applying Lens Distortion

Adherent raindrops on a windshield cause a lens distortion, similar to the fisheye or barrel effect. This distortion can be represented as a nonlinear spatial translation of image points into the raindrop pixels. This translational transformation can be approximated by [73],

$$P_n = P_o + DF * P_o^3 \quad (6.1)$$

where $\boldsymbol{P_n}$ is the distorted pixel in the raindrop, $\boldsymbol{P_o}$ is the original (environment) pixel that is influenced by the raindrop distortion, and $\boldsymbol{DF}$ is the distortion factor. We use the MATLAB function "geometricTransform2d" to represent this lens distortion effect of a raindrop. Figure 6-5 (a) shows the distorted region after applying the lens effect.

6.3.3 Blurring, Resizing, Rotating

Since the vehicle camera used in vision-based applications is usually focused on the environment, any close images, raindrops included, would look blurry [74, 30].

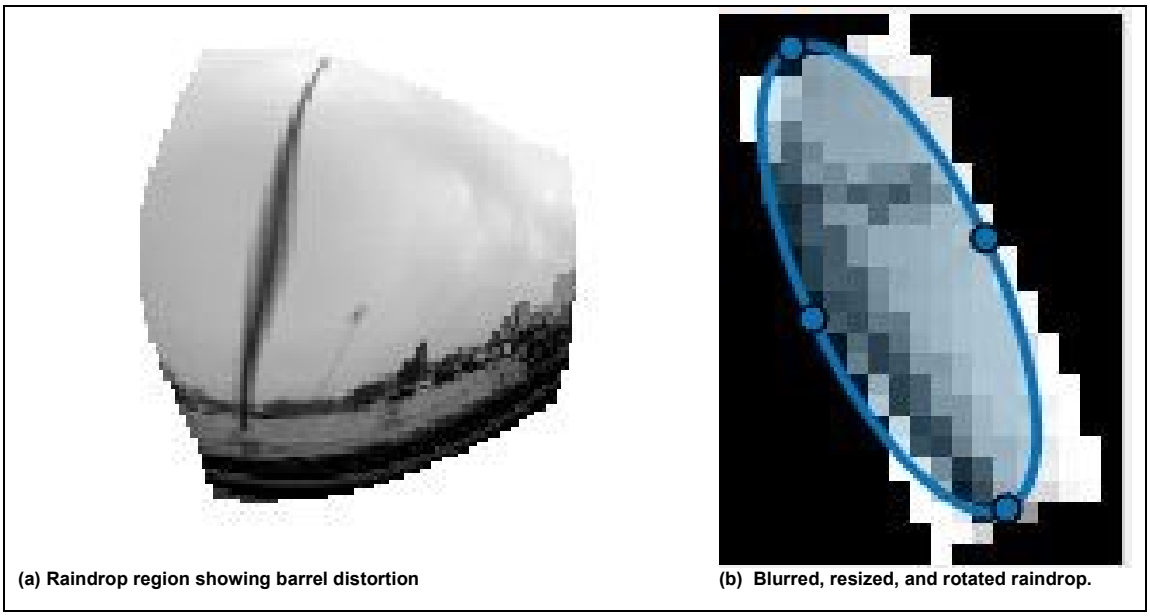


Figure 6-5: Applying translational transformation on an image produces the barrel effect (a). The distorted region is blurred, resized, and rotated to match desired raindrop characteristics (b).

We used the MATLAB function “imfilter” to add the blurring effect to our simulated raindrops. For focus-blurring, we selected the correlation option and set the blur window size to a proper value. For motion-blurring, we used the “fspecial” function to create a special filer type, with the ‘motion’ option, and the X and Y motion-blurring levels set appropriately. This motion type is then used by the imfilter function to add a motion-blurring effect. The parameters for focus and motion blurring were determined experimentally. We then resize and rotate the raindrop image, to approximately match the encapsulating ellipse we have started with. Figure 6-5 (b) shows the raindrop region, after blurring and applying the resize and rotation operations.

6.3.4 Adding Raindrop to Image

Adherent raindrops tend to be slightly brighter than their surrounding background, since they collect light from all areas of the image, due to the lens effect. As stated earlier, raindrops lack strong boundaries that separate them from their background and give them specific shapes. We use intensity adjustment and border dilation and filtering to allow for seamless addition of generated raindrops to the original (clear) image. Figure 6-6 shows samples of generated raindrops compared to real raindrops in a wet image.

6.3.5 Capturing Adherent Raindrop Dynamics

Raindrops remain adhered to the windshield surface so long as the forces exerted surface tension and gravitational pull are balanced. You et al. [74] found that the observed raindrop speed was around 0.01 pixel/s, as seen by a camera mounted on a vehicle moving at a speed of 30 km/h. They also observed that the motion seen inside a raindrop was 20 to 30 times slower than that seen in other areas of the image. In our raindrop simulator, the raindrop dynamic behavior is implemented as follows:

1) No movement is applied to raindrops from one frame to the next, a reasonable approximation to the quasi-static movement observed by You et al. [74] .

2) New Raindrops are added arbitrarily to the raindrops generated on previous image frames.

(a) Simulated **(b) Real**

Figure 6-6: Generated raindrops are added to a clear image (top) that matches real raindrops of the same scene, captured under rainy conditions (bottom). Each raindrop pair (real, generated) is encapsulated with an ellipse of the same color. Real and generated raindrops are visually very similar, as perceived by a human observer.

3) If a new raindrop is generated that intersects with an existing one, the distorted area is generated as a simple addition of the two raindrops. This method allows for approximating complex raindrop shapes as a combination of elliptical shapes.

4) For simplicity, the process of refactoring large raindrops to smaller droplets (see Stuppacher and Supan [32]) is not implemented. This simplification holds reasonably well under light-to-moderate raindrop intensity since the size of raindrops does not grow fast, due to the low probability of arbitrary raindrops intersecting over a short period.

5) Raindrops' mask is refreshed (all raindrops regenerated) every 20 to 30 frames (programmable), to account for the dynamic changes of background scene elements and, at the same time, making use of You et al. [74] observation about the slow change of raindrop pixels compared to non-raindrop areas.

6.4 Results and Analysis

To validate the quality of our generated raindrops against real ones, we started with a clear/rained image set of the same scene. We individually picked raindrops from the rained image and measured their positions, sizes, and orientations. We then used our simulator to generate raindrops with the same characteristics as the real ones. We used SSIM and EMD metrics to measure the level of similarity of our generated raindrops to their real counterparts. We took each rain raindrop image and compared it to the corresponding simulated raindrop, which was generated by using the same orientation, size, and position of the real raindrop image. Figure 6-7 shows the similarity measure histogram between real and simulated raindrops, as calculated using EMD and SSIM metrics. Figure 6-8 shows that the similarity level between a real rained image and an image with generated raindrops increases with the addition of extra simulated raindrops.

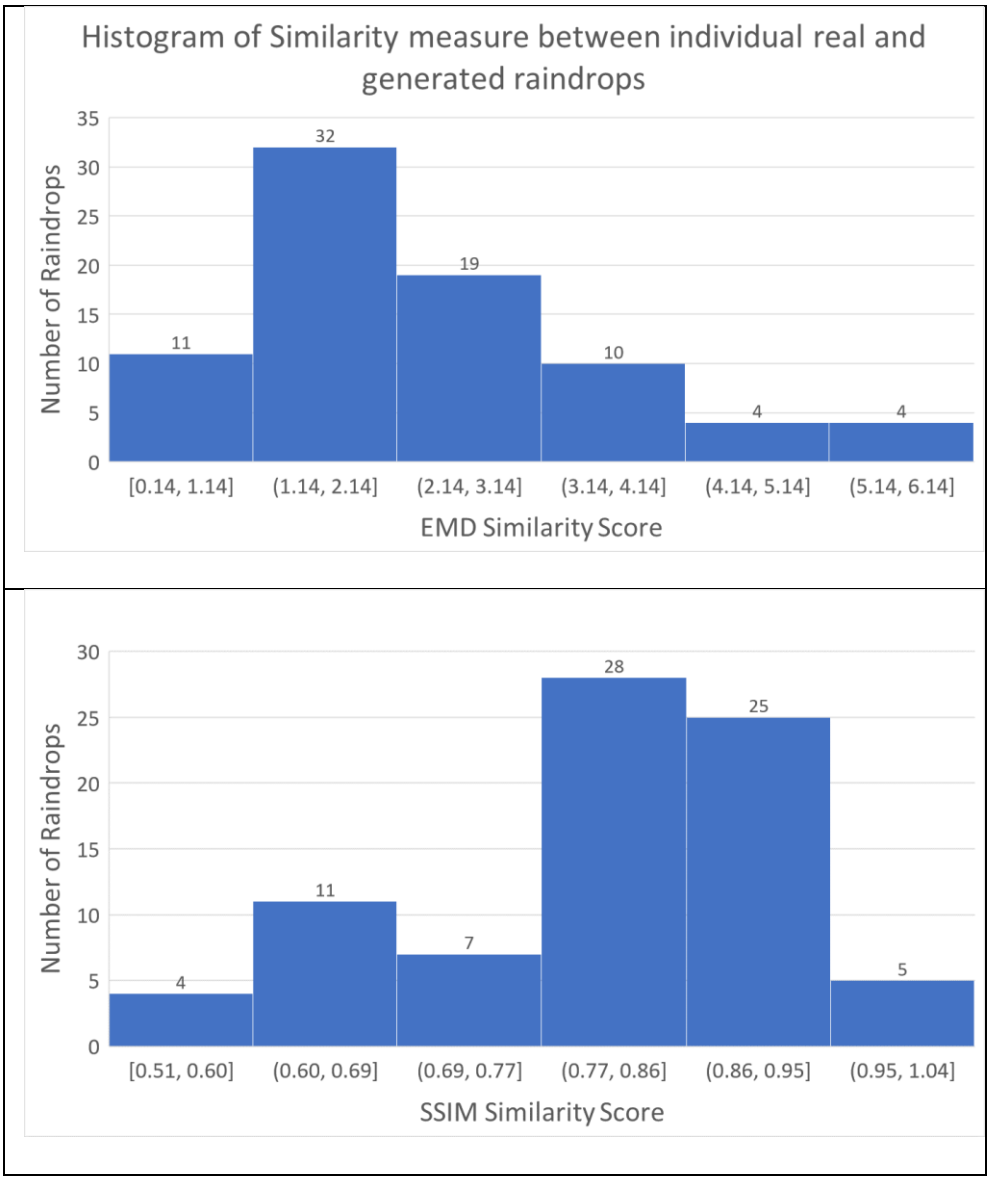


Figure 6-7: Similarity between individual Real and Simulated raindrops is measured using EMD (top) and SSIM (bottom) metrics and the histograms of scores calculated for each metric. The figure shows a strong similarity between the real and generated raindrops

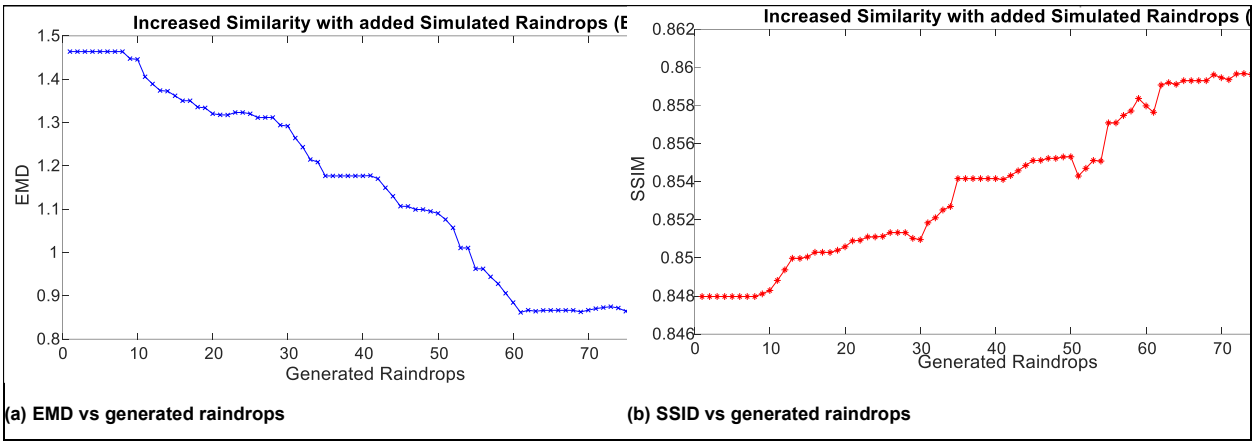


Figure 6-8: Using EMD (a) and SSIM (b) as similarity measures of real rained image and clear image with simulated rain added shows a clear trend towards improving similarity, with the addition of simulated raindrops. Lower EMD scores and higher SSIM scores both mean increased similarity levels between compared images.

In the second level of testing, we generated rained images by adding generated raindrops to clear images. Real and simulated rain image frames are then selected based on the degradation level of each image frame as compared to the clear image frame of the same scene. SSIM and EMD metrics were used as indicators of image degradation, in the sense that a worse similarity score of these metrics was taken as a direct indication of increased image degradation caused by raindrops. Only "parking-lot" data sets were used in this series of tests, to eliminate any degradation from the movement of the test vehicle, relative to other objects in the scene. The matched real and simulated rained images are then used as inputs to three deep learning-based object detectors, namely Single Shot Detector (SSD), You Only Look Once version 3 (YOLOv3), and Faster Region-based Convolutional Neural Network (RCNN). Detected objects are evaluated and matched, and detection performance is evaluated in two ways:

6.4.1 Detection Confidence Level Versus Image Degradation Level

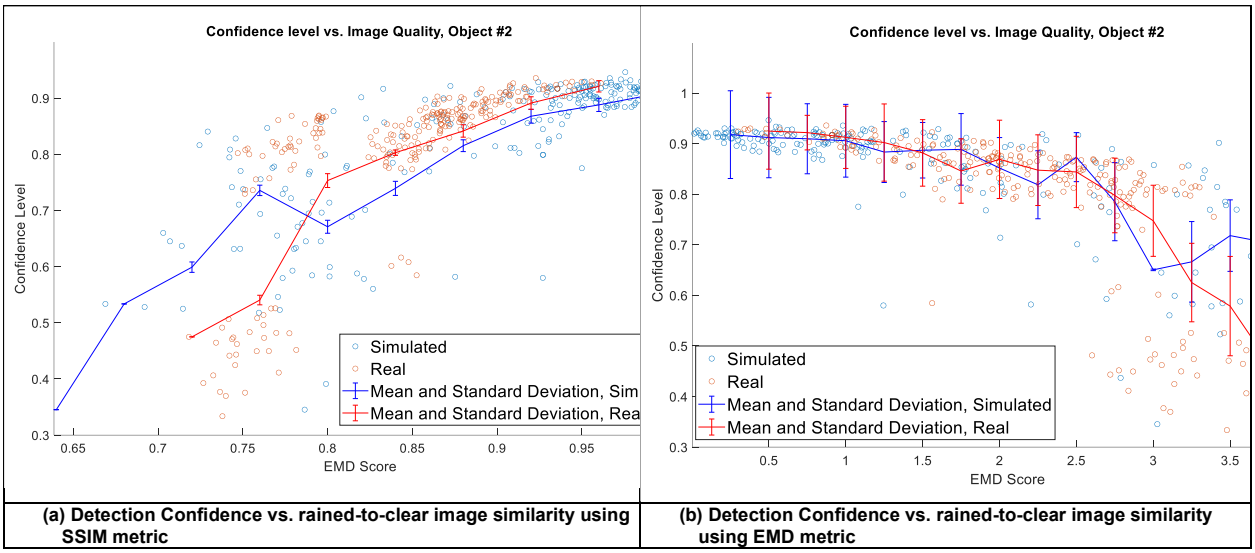
The confidence level that is generated by the object detectors and assigned to each detected object (0-100%), is inspected against image degradation (dissimilarity to the clear image), over

all dataset frames. This process is repeated for both real and simulated rained images. Figure 6-9 shows matched image frame pair of real and simulated rain, with objects detected in each image with different confidence levels. Figure 6-10 shows plots of confidence levels of one object (Object #2) in the image scene, against SSIM and EMD, used as measures of distortion. There is a clear trend of increased confidence level with decreased degradation (less rain) of rained images. This trend is observed in both real and simulated rained images. The error bars represent the mean (center of the bar) and standard deviation (length of the bar) of sample point segments, each segment containing sample points that have the same range of SSIM or EMD scores.

(a) Real rained image

(b) Image with simulated rain

Figure 6-9: Objects are detected in real (a) and simulated rained images (b), with different confidence levels (using YOLOv3). Bigger objects are detected with higher confidence levels than smaller ones. The detectors order the detected objects according to their detection confidence levels.

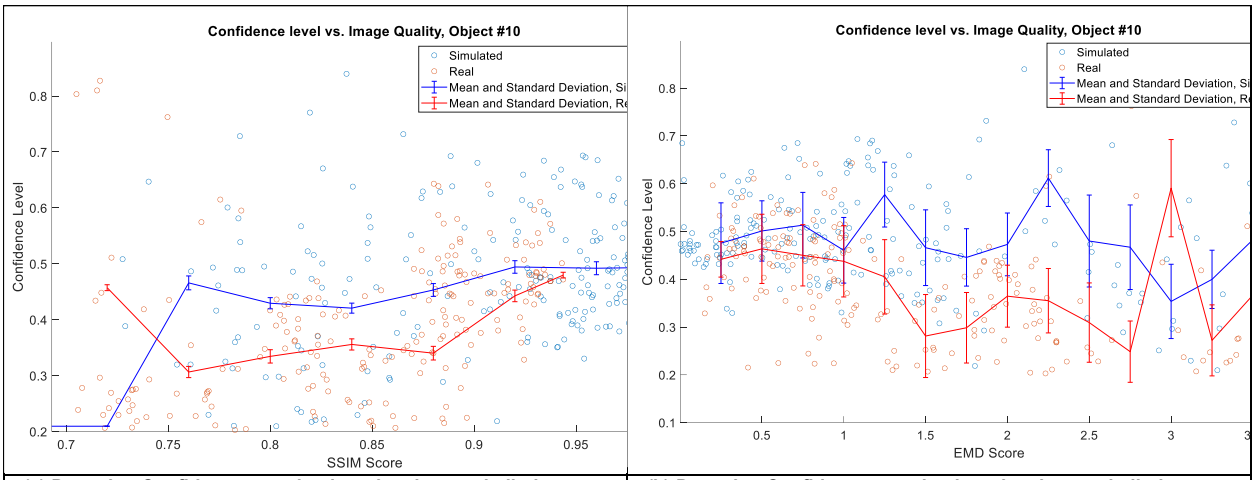


(a) Detection Confidence vs. rained-to-clear image similarity using SSIM metric

(b) Detection Confidence vs. rained-to-clear image similarity using EMD metric

Figure 6-10: Detection Confidence level of Object #2 increases with decreased image degradation in both real and simulated rain images. The mean of sample detection confidence levels (center of error bars) has a strong correlation to image quality.

Figure 6-11 shows a plot of another object (Object #10) detection confidence levels versus image degradation levels. The trend is still visible on both real and simulated rain images but not as strong as the first object. We calculated the correlation between detection confidence and image degradation for several objects in the real and simulated datasets. The results are shown in Table 6-2. As expected, object 2 showed a strong correlation between its detection score and image quality. The correlation scores for real and simulated rained images for object 2 were also very comparable. Object 10, on the other hand, showed a weaker correlation score, which explains why the trend was observed in Figure 6-11. The table also shows that object 1 and object 15 show no clear correlation between detection confidence and image degradation level. Further analysis showed that object 1 was the largest one (car) in the image scenes, and its detection confidence remained high under all levels of image degradation. Object 15 was the opposite. It was very small and its detection confidence was low at all levels of image degradation. In both cases, detection confidence levels were not strongly correlated to image degradation, caused mainly by adherent raindrops.



(a) Detection Confidence vs. rained-to-clear image similarity using SSIM metric

(b) Detection Confidence vs. rained-to-clear image similarity using EMD metric

Figure 6-11: For small objects in the image (e.g., Object #10), the detection confidence level is low, even at low image degradation levels. The correlation between detection confidence and image quality is also weaker than larger and brighter objects in the same image (e.g., Object #2).

Table 6-2: Correlation is calculated between detection confidence and image quality for real and simulated rained images. Comparable correlation scores for real and simulated rained image objects. Some objects show weak to no correlations.

	Correlation between Confidence and EMD		Correlation between Confidence and SSIM	
Object ID	Real	Simulated	Real	Simulated
1	-0.3007	0.3422	0.2536	-0.4207
2	-0.7440	-0.6641	0.7690	0.7390
3	-0.6552	-0.2923	0.6453	0.6167
4	-0.7019	-0.8262	0.6827	0.8427
5	-0.3589	-0.7343	0.4529	0.7421
6	-0.4145	-0.5397	0.4538	0.5034
10	-0.2617	-0.3070	0.2141	0.3390
15	0.3912	-0.0365	-0.2431	-0.3717

Figure 12 shows histograms of correlations between detection confidence and image quality, for fifteen objects in the scenes of the images, calculated for both real and simulated rained image sets.

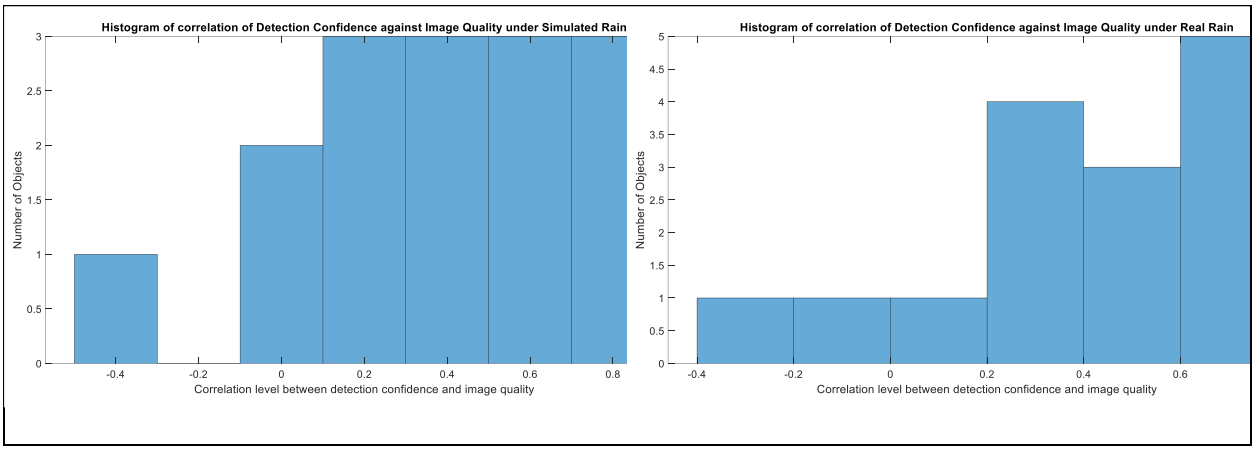


Figure 6-12: Histograms of correlation of detection confidence and image quality for both real (left) and simulated (right) rained images show the strongest correlation levels under both real and simulated rain. Only a few objects had weak correlation, and around half the objects showed relatively strong correlation levels (above 0.5).

6.4.2 Precision and Recall Metrics Versus Image Degradation Level

The other means of assessing the performance of our raindrop simulator is using precision and recall metrics, instead of just confidence levels, against image degradation levels. Initially, we ran detection algorithms on clear image sets and used them as the ground truth for our precision and recall calculations. A detection is considered true positive (TP) if the detected object in the rained image (real or simulated) matched that found in the clear image. A false negative (FN) is considered when an object in the clear image is not detected in the rained one. A false positive (FP) is when the classifications of the objects detected in clear and rained images do not match (e.g., car vs. boat). Figure 6-13 shows the plot of recall against image degradation, represented with EMD measure. As can be seen from Figure 6-13, there is a clear trend of decreased recall scores with the increase of image degradation, represented by the EMD similarity metric. The trend is observed in both real and simulated rained image sets.

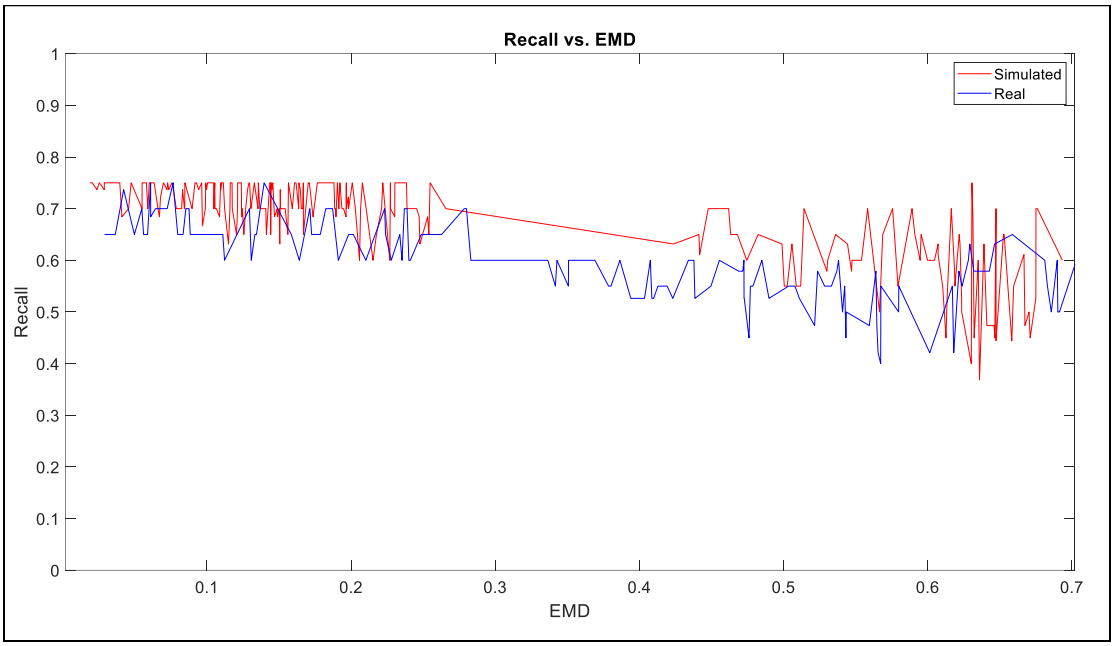


Figure 6-13: Calculating the recall score of detected objects over all captured frames of rained images, with different rain intensities, shows a trend of decreased recall score with increased image quality, represented by the EMD similarity score. As the degradation in image quality increases, objects are detected less often, and recall score correlation to image quality becomes weaker.

Table 6-3 shows the correlation value of recall score versus image quality (EMD and SSIM) for both real and simulated rained sets.

Table 6-3: Correlation is calculated between detection confidence and image quality for real and simulated rained images. Comparable correlation scores for real and simulated rained image objects. Some objects show weak to no correlations.

Recall vs. EMD		Recall vs. SSIM	
Real	Simulated	Real	Simulated
-0.7645	-0.8097	0.7253	0.8356

Precision scores calculated on the same datasets did not show a clear dependency on the degradation levels of rained images. Examining the detection results, we found that the dominant failure mode was false negative rather than false positive. This can be explained as follows. The object detectors were trained with full or partial images of common objects that can be found on the street. A raindrop may occlude sections of an object, but the remaining un-occluded section may still be sufficient features to correctly classify the object. Only when the occlusion is

significantly large enough, that the detector fails to detect (false negative) the raindrop-occluded object. It is much less likely that the occlusion would leave sections of the object, which would cause the detector to classify it incorrectly (false positive). Since precision is calculated as $\frac{TP}{TP+FP}$, it is clear why the precision score came as one for most of the samples, and thus was weakly correlated to the degradation level. The recall, however, is calculated as $\frac{TP}{TP+FN}$ so it was more correlated to the degradation level and showed a significant decrease with the increase of the degradation level.

6.4.3 Comparative Analysis

To evaluate the performance of our raindrop simulator against the state-of-the-art ray-tracing-based raindrop simulators, we used Carlin's [75] model to generate rainy images from clear 500x500 pixel images, that we selected from our original dataset. A total of 128 images with different rain patterns were used. Figure 6-14 shows an example of an image with generated raindrops using our model and Carlin's model. Carlin's model generates raindrops with similar shape and orientation, compared to raindrops generated from our model that vary in size, shape, and orientation. Roser et al. [33] modeled raindrops using Bezier curves and showed that the area of a raindrop as seen on a windshield is proportional to its volume and maximum thickness. For real raindrops, the bigger the raindrop volume is, the less transparent the raindrop becomes. Raindrop transparency level in Carlin's model is also higher than that generated by our model, and higher than what is normal for the size of raindrops generated by his model.

For quantitative comparison, we evaluated the performance of CNN-based object detectors, using rained images generated by both Carlin's model and ours. We used two metrics in our evaluation, detection confidence level, and detection recall score. For the confidence level evaluation, we

matched objects in the rained images that were generated by both models to those detected in real rained images. We then calculated the differences in detection confidence for each object detected in real and simulated rain images. Table 6-4 shows a summary of some statistical metrics for the object detection confidence level, using real and simulated rain datasets The two models seem to produce similar results in terms of object detection confidence levels, as indicated by the mean and standard deviation metrics of the results.

Figure 6-14: Images with raindrops that were generated by the ray-tracing method (left) and our method (right). Our model generates raindrops with more varieties in size, shape, and orientation compared to the ray-tracing model. The transparency levels in our generated raindrops are closer to that of real drops and are generally lower than that of raindrops generated by the ray-tracing model.

Table 6-4 shows a summary of some statistical metrics for the object detection confidence level, using real and simulated rain datasets The two models seem to produce similar results in terms of object detection confidence levels, as indicated by the mean and standard deviation metrics of the results.

Table 6-4: Mean and standard deviation of object detection confidence levels show statistical similarity of results under real and simulated rain datasets.

	Statistical metrics of detection confidence results	
Datasets	Mean	Standard Deviation
Real rain dataset	0.8029	0.1702
Our generated rain dataset	0.8013	0.1852
Ray-tracing generated dataset	0.8108	0.1857

For the Recall score metric, we matched the image objects detected in simulated rained images from the two models, to the ones detected in the clear image (reference) dataset. Recall score is calculated for each image frame and the results are compared to the recall score of detection with real rained images. Table 6-5 shows a summary of some statistical metrics for the object detection recall score, using real and simulated rain datasets. The object detection recall scores are closer for our model to those with real raindrops than the scores calculated for the ray-tracing model.

Table 6-5: Mean and standard deviation of object detection recall scores show statistical similarity of results under real and simulated rain datasets.

	Statistical metrics of detection Recall results	
Datasets	Mean	Standard Deviation
Real rain dataset	0.6484	0.1956
Our generated rain dataset	0.7601	0.1742
Ray-tracing generated dataset	0.8132	0.0864

6.5 Conclusion

Our proposed simulator generated a visually convincing adherent raindrop on a vehicle windshield. The model performs best when generating simple raindrops that can be approximated with an ellipse. For more complex raindrop shapes, the model can be programmed to generate several intersecting elliptical raindrops, each approximating one section of the complex raindrop shape. This technique was tested by trying to mimic real raindrops of complex shapes using our simulator. Results showed great improvement of raindrops similarity, compared to using a single ellipse

representation of complex shapes. The object detection tests we conducted using three CNN-based deep learning object detectors showed similar behavior using real or simulated rained datasets. This "behavior" can be described as follows:

1) The correlation values between recall score and image quality were very close on all datasets tested and using both YOLOv3 and Faster-RCNN detectors
2) The correlation values between detection confidence levels and image quality were also close on all datasets and the same detectors.
3) Big objects showed Resilience to raindrop-induced image degradation, and that behavior was similar in both real and simulated rained image datasets. Smaller objects in the image were more susceptible to the presence of raindrops and this susceptibility was similarly observed in both real and simulated rained datasets.

EMD and SSIM were good metrics for evaluating degradation in image quality at different levels of raindrop content in an image. They, however, are not perfect. Special attention needed to be applied to limit the influence of dynamic background objects, whether being a distant vehicle, moving clouds, or even flickering street lights. We also observed that they do not always agree when representing image similarities, in a sense that increased SSIM score does not always mean a decrease in EMD score, for the same sets of images compared. This meant that these two metrics cannot be used interchangeably for individual image matching. For observing trends that extend over many samples, the metrics show similar behavior and they appropriately track the progression of image degradation, caused by increased raindrop presence.

Comparison of rained images generated by the state-of-the-art ray-tracing-based model showed very close results, both in visual perception or the generated raindrops, and the usability of generated rained images in object detection system validation.

In terms of performance speed, we developed our raindrop simulator using MATLAB 2018b scripting language, with no specific optimizations. We ran it on a PC with an AMD FX-8350 microprocessor, 16 GB of DDR3 RAM, a 500 GB SSD hard drive, and running Windows 10 operating system. It took on average 600 ms to generate each raindrop, using the full (1280 x 650) image as an input. Figure 6-15 shows samples of our generated raindrop images, alongside the original, clear images, and real rained images with roughly the same level of rain-caused degradation, as our generated ones.

Figure 6-15: Examples of Clear, Real, and randomly generated raindrop images from our dataset. rain intensity ranges from light (set 1) to relatively heavy (set 4). The generated raindrops are perceptually convincing to a human observer.

Chapter 7

Improving the Performance of Automotive Vision-based Applications Under Rainy Conditions

7.1 Introduction

Automotive systems including vision-based applications are highly regulated and are required to meet high performance and safety standards. This means that these systems must operate under all conditions, favorable or adverse. The quality of the system inputs has a direct impact on its performance, in the sense that noisy inputs usually result in degradation in system performance. Two approaches are usually implemented to reduce the effect of noisy inputs on system performance, denoising the inputs, or reducing system sensitivity to noise. Filtering analog signals and debouncing digital ones are two examples of common input signal denoising techniques. Predictive modeling and sensor fusion are system design techniques that lead to reduced system sensitivity to noisy inputs. Rain is a type of adverse weather condition that degrades the quality of images and the performance of vision-based systems that consume them. In our research work [24], we showed that the performance of state-of-the-art object detectors (including YOLOv3, RCNN, and SSD) greatly degrades when applied to image sets with adherent raindrops content in them. Test results showed the drop in performance of the tested object detectors was as high as 77%, as measured by the total number of objects detected and the recall metric.

The majority of research work (see, for example, [28, 37, 45]) is focused on denoising the rain-degraded input images to vision systems, by removing rain content from these Images.

As we have shown in our survey paper on adherent raindrop removal techniques [25] , none of the reviewed de-raining algorithms could perfectly restore the rained images to resemble the clear ones. Classical de-raining techniques use some set of physical properties, such as raindrop shape, size, moving speed to create the raindrop detection model [33]. Other algorithms are based on the optical properties of the raindrop, including its reflective and refractive behavior [27], and its color and texture properties [35]. Spatio-temporal analyses are sometimes added to improve the detection quality of raindrops [40].

The improvements in deep-learning and convolution neural networks (CNN) [45, 53, 54] opened the door for a new set of de-raining techniques that, generally, achieved better performance levels compared with classical machine learning algorithms.

CNN models, however, require large sets of data for training. For some de-raining algorithms, an accurate mask of raindrops is needed to train the CNN model. This requires a large set of matched clear and rained images to generate such a mask. Constructing such a dataset of paired images is not an easy task, due to the unpredictability of rain and the background objects, and due to the variations of the raindrop sizes, shapes, and orientations.

7.2 Method

We propose a different approach to improving vision-based system performance under rainy conditions. Rather than denoising (de-raining) the input images, we propose to reduce the system's sensitivity to noisy inputs. This can be achieved by retraining models that are already trained with clear image sets, with matching sets of rained images. This approach eliminates the need for developing and training the de-raining network. Furthermore, retraining networks

designed for common automotive vision applications (e.g., traffic sign recognition, object detection, lane detection) is efficient and fast, since it employs transfer-learning, whereas training a de-raining network may need to be done from scratch. Table 7-1 shows some differences between the input denoising approach and the network retraining one.

Table 7-1: Differences between input denoising and network retraining approaches for improving vision system performance

Performance boost approach **Comparison points**	De-raining of input images	Retraining model with rained images
Training type	Training from scratch	Transfer learning
Training dataset size	Large	Relatively small
Input type	(Clear, Rained) pair dataset plus raindrop mask and/or structure or texture maps	(Clear, Rained) pair dataset
Objects of interest	Natural raindrops with weak borders and variable shapes, sizes, and orientations	Man-made objects with strong boundaries and uniform shapes (e.g., Vehicles, traffic signs, road marks)

To test our hypothesis, we trained an object detector and semantic segmentation models a clear image set, then retrained it with generated raindrops dataset. A comparison of the detector's performance with clear, rained, and de-rained images showed that the retraining approach showed better performance improvement than the de-raining approach.

7.3 Data Collection and Data Preprocessing

We used different datasets for training and testing the object detection network and for training and testing the image semantic segmentation network.

7.3.1 Object Detection Datasets

We used the 2d "Object Detection Evaluation" from the KITTI Vision Benchmark Suite [61] to train the baseline Yolo3 for detecting objects under rain-free conditions[1]. This dataset consists of training and testing datasets, but we have used the training dataset for both training and testing since it comes with object label text files. The dataset includes 7482 color images with common objects encountered in a drive cycle shown in the background. We modified the format of the label text files to be compatible with MathWorks's deep-learning object label format. The five object classes we chose for the baseline were 'Pedestrian', 'Truck', 'Car', 'Cyclist', and 'Van'.

We collected our own dataset of paired clear and rained images, captured under different driving conditions and showing common road objects in the background. We used the (ELP-960P2CAM-V90-VC) dual-lens stereo camera that was positioned approximately 10 cm away from the windshield. We used the wiping event as a trigger to capture rained and clear image pairs where the frame before the wipe event was captured as the rained image and the frame after it as the clear image. We selected 1162 image pairs to construct our dataset, based on the number of background images in the frames and the degree of similarity between clear and rained image pairs, looking for the highest values in both cases. We then used MALTLAB's '**Image Labeler'** app to label objects in the clear and rained image sets. We chose 'Pedestrian', 'Truck',

[1] KITTI Vision Benchmark Suite is one of the leading vision benchmark suites, that is constantly being updated to match the latest improvements in vision research. It incudes datasets to support research on stereo vision, scence-flow, depth prediction and completion, odomerty, object detection, and other vision research domains.

'Car', and 'None' as the classes for the retrained Yolo3 network. The selection of different object classes from the baseline model was intentional since we wanted to mimic a real-life scenario where transfer learning is used to retrain a baseline network, using different training data and for a different desired output. We used the rained images from our dataset to test the performance of the retrained Yolo3 object detector.

We applied a state-of-the-art de-raining algorithm that was developed by Quan et al. [46] [76] on the same rained dataset, to create a de-rained data set from our rained one, and used it for performance comparison analysis. Quan's de-raining algorithm requires a set of 'edge' images that are generated from the rained image[2]. Figure 7-1 shows image samples from the different datasets we used in the object detection training and testing. There are other publically available implementations of other DNN-based algorithms, including the implementation of Qian et al. [11] [77], and Yasarla and Patel [78] [79].

We chose Quan et al. [76][3] implementation since, first, it was an improvement over Qian's algorithm for image de-raining, given that Qian's algorithm [11] is becoming the new standard of adherent raindrop deraining. Quin et al. [77] also reported de-raining results that surpassed other DNN-based algorithms, including Eigen et al. [45] and Isola et al. [80]. Yasarl and Patel's algorithm was developed for de-raining of falling rain streaks from images. As shown by Peng et al. [53], these rain streak removal algorithms do not yield satisfactory results compared to the

[2] We should mention that Quan's algorithm was trained tested a publicly available dataset that was created by Qian et al. [89]. We did not use this dataset in our analysis, since most images did not include enough background objects that could be detected by the object detector. In addition, the dataset was created with synthesized rain, rather than real rain, by spraying waterdrops on a glass surface in front of the camera.

[3] To avoid any issue that may stem from inaccurate implementation the de-raining algorithm Proposed by Quan et al. [48], we used their implementation of that algorithm [88] with no modifications.

ones designed for adherent raindrop removal, even when they retrained those algorithms on the same adherent raindrop datasets used to train the adherent raindrop removal ones[4].

(a) KITTI Object Dataset

(b) Our Dataset - Clear

(c) Our Dataset - Rained

(c) Our Dataset – De-rained

Figure 7-1: Image samples from the different datasets we used in our research work. The KITTI dataset was captured under clear weather conditions, whereas our dataset was captured under rainy conditions.

[4] The falling rain streaks and adherent raindrops are two different problems in terms of type of degradation they cause to input images. The characteristics (features) of rain streaks and adherent raindrops which the DNN system uses for learning are also different. it is not surprising based on the above that retraining a rain streak removal DNN on adherent raindrop datasets does not yield satisfactory results.

7.3.2 Image Segmentation Datasets

For the image segmentation, we used the "Semantic and Instance Segmentation Evaluation" dataset from the KITTI Vision Benchmark Suite [61], to train the baseline image segmentation network. The dataset consists of 200 images of street scenes, taken under clear weather conditions. Pixel-level color and gray-scale segmented images and instance-level segmented images are also included in the dataset. We grouped the 35 segmentation labels that the KITTI dataset provided, into six labels, 'Sky', 'Vehicle', 'Person', 'Background', 'Road', and 'Unlabeled'. We used the clear images and the color pixel-level segmented images to train the segmentation network. We added generated rain at different levels to the original dataset, to create three synthetic raindrop datasets, Low_Rain, Medium_Rain, and Heavy_Rain. We Structural Similarity Index (SSIM) as an indicator of the amount of rain-induced image degradation. We used a raindrop simulator model that we had previously developed [81] to added generated raindrops to clear images, to create the Low_Rain, Medium_Rain, and Heavy _Rain datasets. An overcast effect was added to simulate real rain lighting conditions since the original KITTI dataset was captured under clear conditions. To do this, the color image is first split into its Red, Green, and Blue channels. The mean intensity for each color channel is then calculated, and the pixel intensities are remapped into a tighter intensity range around the mean intensity. This effectively reduces the color content for each channel, a natural consequence of reduced illumination under overcast conditions. In addition, the recombined color image from the three-channel images has a reduced overall intensity, as a result of reduced intensities in each of its color channels. The final overcast image looks darker and less color-rich than the rain-free one. We used the first two generated rain sets to retrain the segmentation network and the last

one (Heavy_Rain) for testing. Figure 7-2 shows examples of the image datasets we used to train and test the segmentation network.

(a) KITTI clear image DataSet

(b) Color pixel-level segmented image

(c) Training image dataset with low-medium raindrops content

(d) Testing image dataset, with High raindrops content

Figure 7-2: Datasets used for training and testing the image segmentation network. KITTI Semantic and Instance Segmentation Evaluation dataset, (a) and (b) is used to train the baseline segmentation network. We added an overcast effect and generated rain to the image sets in (c) and (d) to train and test the segmentation network under rainy conditions.

Table 7-2 shows a summary of the datasets we used in the object detection and image segmentation networks.

Table 7-2: A list of the datasets used in our research for training and testing the object detection and segmentation networks

Set ID	Usage
KITTI_Objects	Train the baseline detector using the KITTI dataset.
Clear_Objects	Retrain the baseline detector using our rain-free dataset
GeneratedRain_Objects	Retrain the baseline detector using our generated-rain dataset
RealRain_Objects	Test the baseline and retrained detector under real-rain conditions.
Derained_Objects	Test the baseline detector using de-rained.
KITTI_Segmentation	Train the baseline segmentation model using the KITTI dataset.
GeneratedRain_Segmentation_Train	Retrain the baseline segmentation model using generated-rain dataset
GeneratedRain_Segmentation_Test	Test the retrain segmentation model using generated-rain dataset
Derained_Segmentation	Test the baseline segmentation model using de-rained images

7.4 Models Training Process and Testing

In this section, we will describe the training process and test cases we conducted for the object detection and semantic image segmentation models.

7.4.1 The Object Detection Model

7.4.1.1 Baseline Model Setup and Training

We used MathWorks's Yolov3 object detector example [82] as our starting model. The detector was based on SqueezeNet [83] Deep Neural Network (DNN). This SqueezeNet, as the name implies, has a relatively small architecture but it produces accurate results comparable to much larger DNNs, such as AlexNet [84]. This allowed us to conduct all our training and testing on a desktop with relatively outdated specifications (AMD FX-8350 with 16 GB of DDR3 RAM and

an Nvidia 1050Ti GPU). Figure 7-3 shows the training stages and datasets used in each stage for the YOLOv3 object detector model.

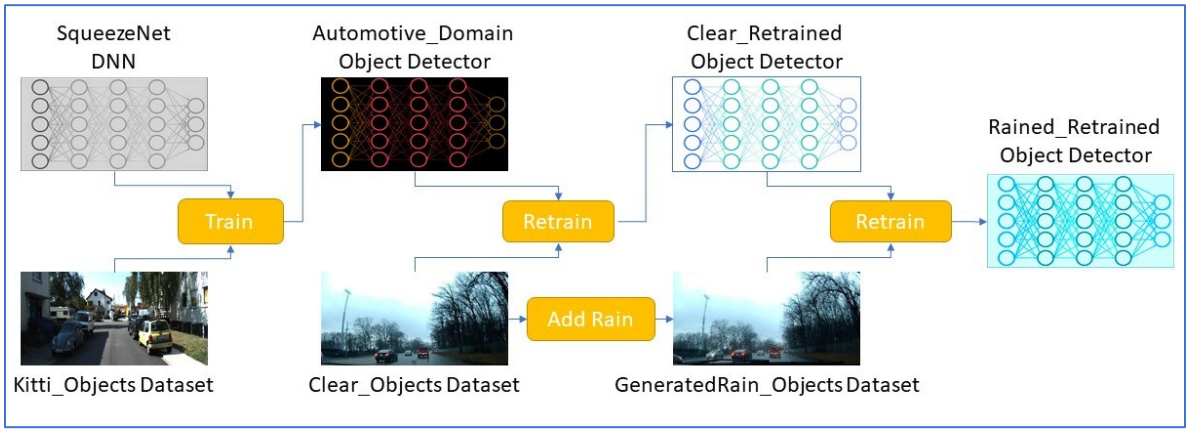


Figure 7-3: A Flow diagram showing the different YOLOv3 model training stages and the training dataset used in each stage

The training process is described below.

7.4.1.2 Train the Automotive-specific Object Detector

We trained the starting model using the "KITTI_Objects" dataset, split as 70% training and 30% testing. The maximum number of epochs was set to 200, with a minimum batch size of 8 and a maximum learning rate of 0.001. We used image augmentation to increase input dataset size, without actually adding more images to the training dataset. We used six anchors to improve image object fitting. Both data augmentation and anchor box calculation functions are part of MathWorks's YOLOv3 model. Table 7-3 shows the statistical results of testing the resultant object detector using the remaining 30% of the "KITTI_Objects" dataset. Some metrics commonly used for detection performance assessment are the Average Precision (AP) and Log-Average Miss Rate (LAMR). In MATLAB, the function "evaluateDetectionPrecision" can be used to calculate the AP score, which was based on the PASCAL VOC2011 [20] definition of AP. To calculate the LAMR score, MATLAB provides the function "evaluateDetectionMissRate" which is implemented based on Dollar et al. pedestrian detection

evaluation paper [85]. Figure 7-4 shows an example image from the test dataset with detected objects annotated.

Table 7-3: The average precision and log-average miss rate scores, as calculated for the five object classes in the automotive-domain object detector. Larger average precision scores and smaller log-average miss rate scores are desirable for better detection performance.

Object Class	Average Precision	Log-Average Miss Rate
Pedestrian	0.59	0.45
Truck	0.90	0.08
Car	0.81	0.37
Cyclist	0.64	0.34
Van	0.81	0.19

Figure 7-4: An example of the output of the Yolov3 detector that was trained in stage 1. The objects are identified with a bounding box, with a class tag and detection confidence level shown for each object.

7.4.1.3 Train the Rain-free Object Detector

In this stage, we used the "RealRain_Objects" dataset to retrain the Yolov3 network from the previous stage, to detect three different object classes, 'Pedestrian', 'Truck' and 'Car'. Through the power of transfer learning, we managed to retrain the object detector with very little change to the actual DNN structure. Since the dataset size in this stage is smaller than the one used in the previous stage, we had to run the training process for 300 epochs but we kept all other training parameters the same. We then tested the retrained detector using the "RealRain_Objects" real

rainy image set, to evaluate the detection performance degradation due to the presence of raindrops. We also tested the retrained object detector on the "Derained_Objects" dataset, to evaluate if there were any performance improvements using de-rained images versus rained ones.

Table 7-4 shows a summary of the AP and LAMR performance metrics for the three object classes using rain-free, rained, and de-rained images. Figure 7-5 shows an example image of object detection at this stage. As expected, the detection performance of the YOLOv3 detector that was trained on a clear image degraded considerably with rained image set used as an input. This is indicated in both decreased AP scores and increased LAMR scores for all three object classes. In addition, it seems like the de-raining process degraded the detection performance even further than the performance under the original rained images.

Table 7-4: The average precision and log-average miss rate scores, as calculated for the three object classes in the rain-free object detector. As shown in the table, there is a big degradation in detection performance when using rained images, and an even larger degradation when de-rained images are used.

	Rain-free		Rained		De-rained	
Object Class	Average Precision	Log-Average Miss Rate	Average Precision	Log-Average Miss Rate	Average Precision	Log-Average Miss Rate
Car	0.92	0.09	0.36	0.63	0.18	.81
Truck	0.94	0.11	0.73	0.46	0.57	0.65
Pedestrian	0.64	0.36	0	1	0	1

Figure 7-5: An example of the output of the Yolov3 detector that was trained in stage 2 using clear, rained, and de-rained datasets. The objects are identified with a bounding box, with a class tag and detection confidence level shown for each object. Not much rain content was removed by the de-raining algorithm and no detection performance improvement in the de-rained image compared to the rained one.

7.4.1.4 Train the Rained Object Detector

For this stage, we used the "GeneratedRain_Objects" dataset to retrain the YOLOv3 object detector that we had trained in the previous stage. This dataset represents simulated rainy conditions, where generated raindrops are added to the clear images in the "RealRain_Objects" dataset. We then tested the retrained detector using the "RealRain_Objects" real rainy image set. Table 7-5 shows the AP and LAMR performance metrics for the three object classes under rained conditions. The detection performances for the 'Car' and 'Truck' classes were on-bar with those reported with the detector that was trained and tested with a Rain-free image dataset. In other words, retraining the rain-free object detector model with simulated rained images allowed it to overcome the raindrop-related image degradation, and perform at levels comparable to those under rain-free conditions. The detection performance for the Pedestrian class is still very low (AP≈0, LAMR≈1). This is because there are much fewer instances of Pedestrians in the dataset than cars and trucks. The AP metric is calculated as the area under the curve that represents the Precision-to-Recall relation. Similarly, the LAMR is calculated as the area under the curves that represent the mapping between Miss Rate (MR) and False Positive Per Image (FFPI) metrics. This type of calculation is useful since it represents the entire curve (Precision/ Recall or MR/FPPI) by a single reference [85]. It does seem, however, to penalize classes with low-occurring instances in the form of very low AP and very high LAMR scores.

Table 7-5: The average precision and log-average miss rate scores, as calculated for the three object classes in the rained object detector. As shown in the table, this retrained detector seems to perform as well as the rain-free detector that is trained and tested on rain-free images.

Object Class	Average Precision	Log-Average Miss Rate
Car	0.91	0.06
Truck	0.95	0.08
Pedestrian	0	1

To verify that the retraining with simulated rain did not affect the performance of the object detector under clear rain conditions, we tested the rained detector with the original rain-free dataset, "Clear_Objects ". As shown in table 7-6, there is no change in performance between the rained detector and rain-free detector, both tested with rain-free images.

Table 7-6: The average precision and log-average miss rate scores, as calculated for the three object classes in the rained object detector. As shown in the table, this retrained detector seems to perform as well as the rain-free detector when tested with rain-free images.

Object Class	Average Precision	Log-Average Miss Rate
Car	0.92	0.09
Truck	0.94	0.11
Pedestrian	0.64	0.36

7.4.2 The Image Segmentation Model

7.4.2.1 Baseline Model Setup and Training

We used MathWorks's semantic segmentation example [86] as our starting model. The example describes the process to train Deeplab v3+ [87] which in the MATLAB example was based on a pre-trained Resnet-18 network [88]. Figure 7-6 shows the training process flow that we used to train the semantic segmentation model to segment rained images. To create a baseline

segmentation model for the automotive domain applications, we train the Deeplab v3+ with the "KITTI_Segmentation" dataset.

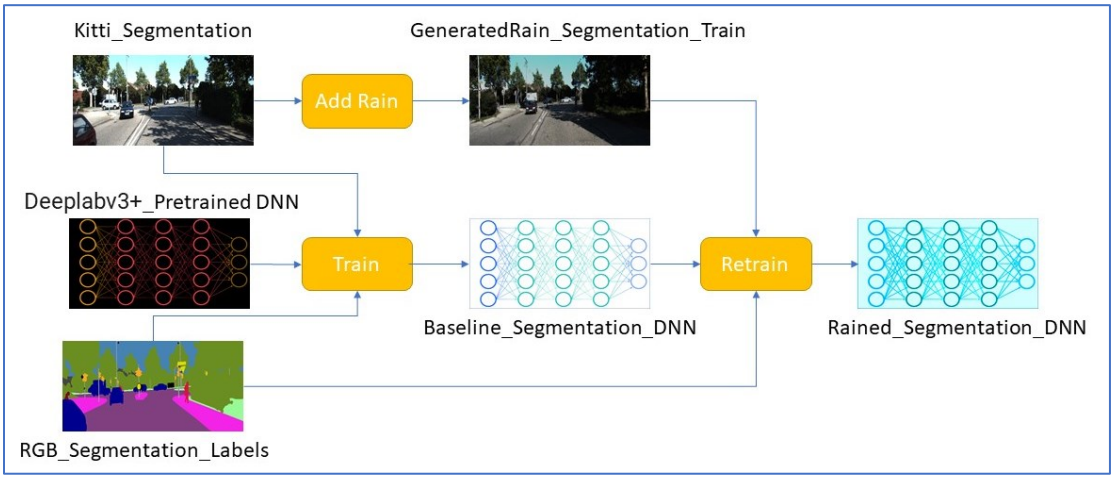


Figure 7-6: The process for training the rained semantic segmentation model. Starting with a pre-trained DeepLapv3+ network, we train the model on a dataset that is more specific to automotive domain applications. We then retrain the segmentation model with simulated-rain images, to improve system robustness to rain-induced image degradation.

We split the dataset as 75% training, 10% validation, and 15% testing datasets. We set the maximum epochs to 300 and the minimum batch size to 8. We set the initial learning rate to 0.001 which is reduced after each concluded epoch. Data augmentation is used to increase the "effective" training dataset size without adding more images. As a common solution to mismatched representations of segmentation classes in the training dataset (i.e., some segments are much more present in the dataset than others), the training weights are adjusted to be inversely proportional to the frequency of occurrence of any given segmentation class. Both functionalities (data augmentation and training weight adjustment) are provided as functions in the MATLAB starting model. The output of this stage is the Baseline_Segmentation_DNN model which we tested using the test part of "KITTI_Segmentation". We used the Intersection over Union (IoU), Accuracy, and MeanBFScore quality metrics to evaluate the quality of segmentation provided by the model. MathWorks provides a good explanation to IoU, Accuracy, and MeanBFScore quality metrics as follows [89]

Accuracy is the ratio of correctly classified pixels in each class to the total actual pixel in that class. Using the True Positive (TP), and False Negative (FN) numbers, Accuracy can be given as:

$$Accuracy = \frac{TP}{TP + FN} \tag{7.1}$$

"IoU is the ratio of correctly classified pixels to the total number of ground truth and predicted pixels in that class" [89]. Using TP, FN, and False Positive (FP) numbers, IoU can be given as:

$$IoU = \frac{TP}{TP + FP + FN} \tag{7.2}$$

MeanBFScore is a measure of the mean Boundary F1 (BF) which indicates how well the predicted boundary of a given class is aligned with the actual boundary of that class.

In MATLAB, the function "evaluateSemanticSegmentation" can be used to calculate these three metrics in image segmentation applications.

Table 7-7 shows a summary of model performance using the above-described metrics and Figure 7-7 shows the confusion matrix for the different segmentation classes detected by the model. The table shows that the segmentation model performs well for all classes, except the "person" class. This is because this class is much smaller in terms of pixels compared to the others, so it would be more sensitive to any mismatches between predicted and actual, as calculated by the three metrics. The confusion matrix in Figure 7-7 shows a high rate of correct segmentation per class (diagonal cells) versus a low rate of incorrect classifications (off-diagonal cells).

Table 7-7: The Accuracy, IoU, and MeanBFScore segmentation quality metrics are shown for the classes that are identifiable by the baseline model across all images in the rain-free test dataset

	Accuracy	IoU	MeanBFScore
unlabeled	0.724	0.266	0.417
sky	0.983	0.964	0.929
vehicle	0.967	0.828	0.837
person	0.328	0.153	0.400
background	0.940	0.924	0.923
road	0.958	0.923	0.868

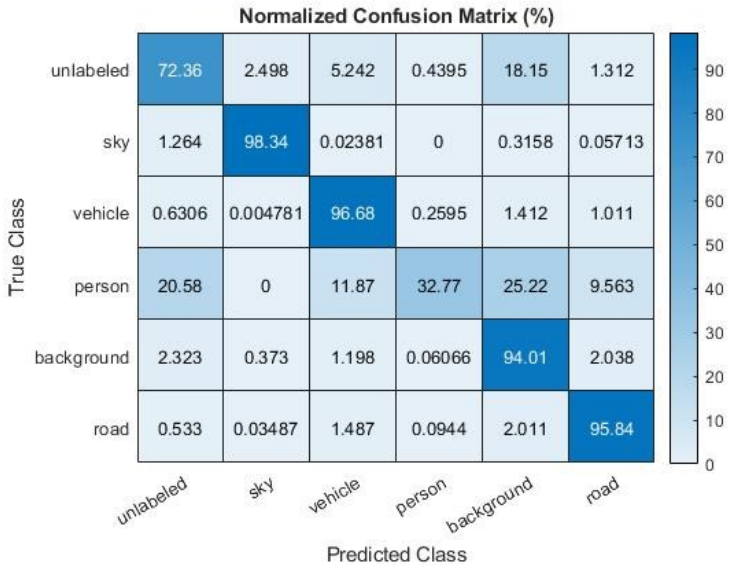


Figure 7-7: The confusion matrix shows the percentage of correct and incorrect segmentation of all classes supported by the segmentation model. The diagonal cells represent the percentage of correct class segmentation, and the off-diagonal cells represent the percentage of incorrect segmentation of the pixels of a given class as belonging to another class.

7.4.2.2 Testing the Baseline_Segmentation_DNN Model with Rained and De-Rained Datasets

To evaluate the effect of rain on the semantic segmentation process, we tested the Baseline_Segmentation_DNN model using GeneratedRain_Segmentation_Test as the input dataset. As described earlier, this dataset contains the simulated rained images with heavy

raindrop content. Table 7-8 summarizes the segmentation performance for each label, using the same statistical metrics as before. Figure 7-8 shows the confusion matrix for the different segmentation classes detected by the model under rained conditions. As expected, the quality metrics show noticeable degradation in segmentation quality when the rain-free segmentation model was used on the rained dataset. We can see from the confusion matrix that the correct segmentation percentage is still much larger than the incorrect segmentation percentage under rained images, except for the “person” class.

Table 7-8: Segmentation quality of the baseline model when tested with the rained image set. Noticeable drop in segmentation quality between rain-free and rained segmentation test, as shown by the three segmentation quality metrics.

Metric / Label	Accuracy	IoU	MeanBFScore
unlabeled	0.100	0.054	0.150
sky	0.84	0.809	0.740
vehicle	0.750	0.463	0.482
person	0.0749	0.046	0.060
background	0.920	0.813	0.825
Road	0.812	0.772	0.677

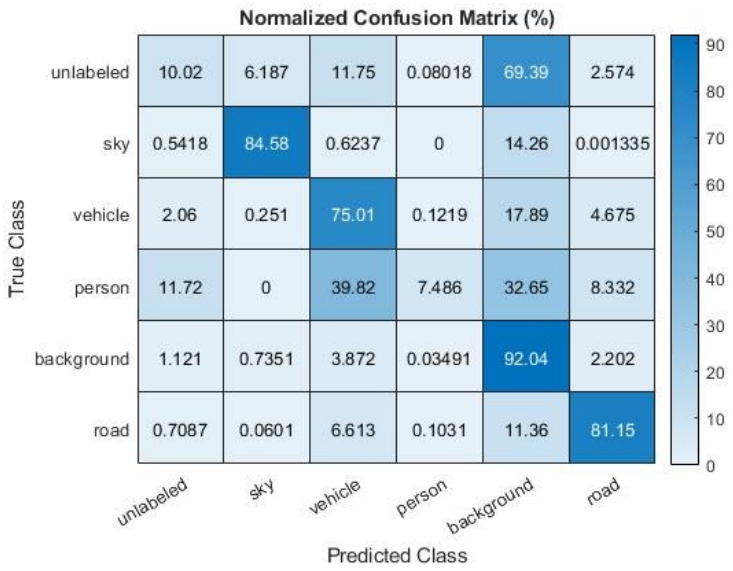


Figure 7-8: The confusion matrix shows a drop in the correct segmentation percentage and an increase in incorrect segmentation percentage across all classes. The "person" class shows the largest percentage drop since its relatively small size makes it more susceptible to the presence of raindrops in the image.

We then tested the Baseline_Segmentation_DNN model using "Derained_Segmentation" as the input dataset. As described earlier, this dataset is generated by feeding the GeneratedRain_Segmentation_Test dataset to Quan et al. [46] de-raining model, to remove raindrops from images. Table 7-9 summarizes the segmentation performance for each label, using a de-rained dataset, and Figure 7-9 shows the associated confusion matrix for the different segmentation classes supported by the segmentation model. The quality metrics show noticeable degradation in segmentation quality when using a de-rained dataset over the original rained one. The largest drop in segmentation quality is observed in the "sky" and "vehicle" classes. The confusion matrix shows that only "road" and "background" classes have a higher correct segmentation percentage than incorrect ones. Another interesting observation is that the majority of incorrect observations are classified as "background" class. The same phenomenon was observed under rain-free and rained segmentation testing which indicates a possible segmentation bias towards the "background" class, even though we used the wights reverse-frequency technique in our design and training.

Table 7-9: The segmentation quality metrics show lower performance of the rain-free (baseline) segmentation model with the de-rained dataset than that under rained dataset. Performance drop was highest for “sky” and “vehicle” classes and the least drop was observed for the “road” class

	Accuracy	IoU	MeanBFScore
unlabeled	0.040732	0.00975	0.055519
sky	0.37414	0.24427	0.41851
vehicle	0.21466	0.1054	0.24246
person	0	0	0.002181
background	0.7009	0.52471	0.68721
road	0.71426	0.57007	0.59807

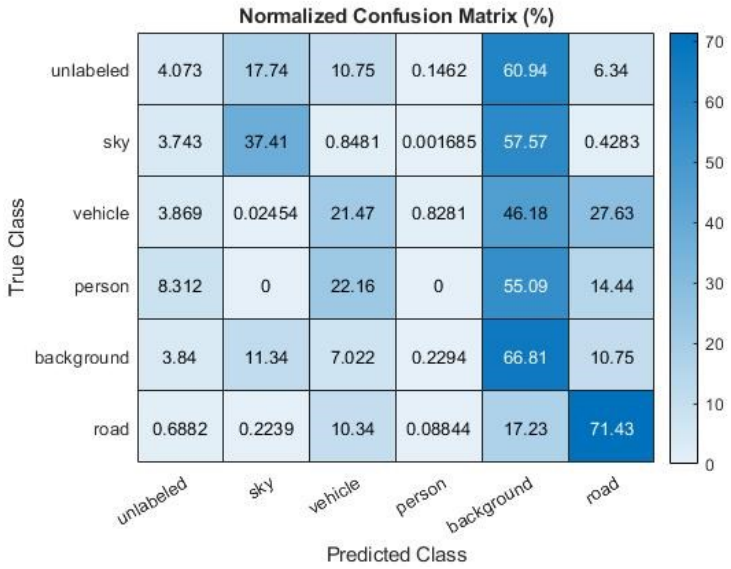


Figure 7-9: the confusion matrix for class segmentation results shows that only "background" and "road" classes still show more correct than incorrect segmentation under de-rained dataset and rain-free segmentation model mix. It also shows that the “background” class contributed to the most percentage of incorrect classifications.

7.4.2.3 Retraining the Baseline_Segmentation_DNN

We retrained the Baseline_Segmentation_DNN model from the previous steps using the “GeneratedRain_Segmentation_Train” dataset. The dataset consists of 400 images with low and medium intensity of generated raindrops added. We split the dataset 75% training, 10% validation, and 15% testing and ran the training process for 200 epochs. All other hyperparameters we left intact from the previous training process. As shown in Table 7-10, there is a big improvement in the segmentation with the rained model compared with the rain-free

model, both tested with real rain image dataset. The confusion matrix in Figure 7-10 shows more correct to incorrect segmentation for each class recognizable by the segmentation model. We then retested the rained segmentation model on the rain-free image set to verify that the retraining with rained dataset did not degrade the segmentation quality for rain-free images. The results shown in Table 7-11 confirm that the segmentation model performance improved with retraining which highlights one unintended benefit from using simulated data for retraining.

Table 7-10: The segmentation performance metrics show that the retrained segmentation model performs on the rained dataset at levels comparable to the performance of the rain-free segmentation model that is tested with the clear dataset.

	Accuracy	IoU	MeanBFScore
unlabeled	0.28243	0.11976	0.22585
sky	0.96257	0.87067	0.8123
vehicle	0.78133	0.54641	0.57202
person	0.3907	0.14859	0.15047
background	0.89598	0.83996	0.84477
road	0.89585	0.83921	0.76265

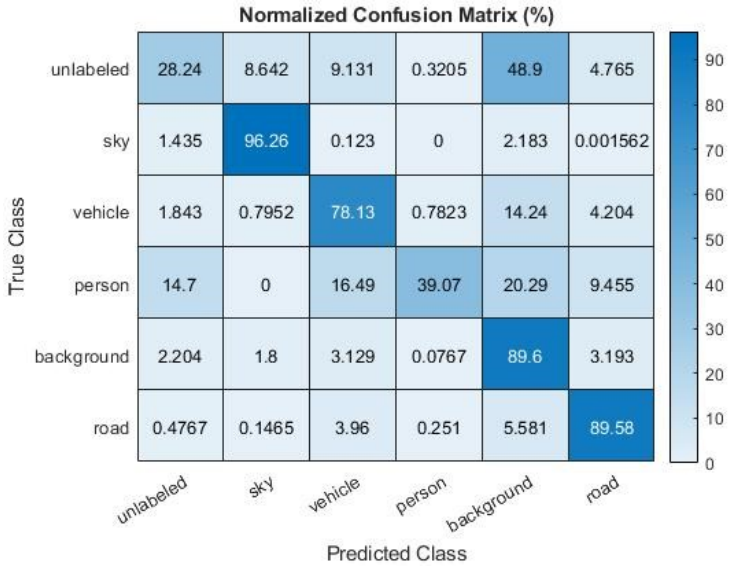


Figure 7-10: Testing the retrained rained segmentation model with a real rain dataset shows that A higher percentage of pixels are correctly segmented for each class than incorrectly segmented.

Table 7-11: Testing the retrained rained segmentation model shows no degradation in performance over the original rain-free segmentation model, both tested on the same rain-free dataset.

	Accuracy	IoU	MeanBFScore
unlabeled	0.92465	0.36152	0.50471
sky	0.98597	0.94929	0.90088
vehicle	0.98732	0.83456	0.80705
person	0.98332	0.3391	0.64998
background	0.93301	0.92611	0.91499
road	0.9565	0.92825	0.88841

7.5 Results and Analysis

We trained a YOLOv3 model to detect common objects encountered in a common drive cycle and tested it using rain-free, rained, and de-rained image sets. The detector performed well on rain-free images, but its performance degraded under rained image set input, as expected. The performance degraded even further for the de-rained image set test, a result we did not expect when we formed our hypothesis. Our results, however, align with the task-driven evaluation results reported by Li et al. [90]. Based on their tests using different object detection algorithms, they concluded that "all existing de-raining algorithms will deteriorate the detection performance compared to directly using the rainy image" [90]. They hypothesized that the de-raining algorithms might need to be optimized to the goal of object detection. This, however, may require a specific de-raining solution to the target vision-based application and consequently reduces the useability and generalizability of the de-raining algorithms.

By analyzing the de-raining algorithm that had been developed by Quan et al. [46], we believe their model was too specific to the training and testing dataset they had used. This made it less useful for the real rain datasets we used in our research, due to the following two factors:

1. Quan's model used a training dataset that used synthetic raindrops for rained images. Real raindrops exhibit more variety in shape and size than the simple droplets formed by spraying water on a glass surface. This likely made raindrop detection harder with real raindrops than synthetic ones.
2. The synthetic dataset used in Quan's model was also taken under optimal lighting conditions which made it easier for raindrops to be detected. The overcast in the background of the real rain dataset, on the other hand, made it harder to identify raindrops by a human observer. This overcast in the real rain dataset likely affected the ability to learn raindrops by the de-raining DNN in [46]

The retrained YOLOv3 model with a simulated raindrop dataset showed great improvement of the rain-free object detector, both tested with the real-rain dataset.

The only class that did not show improved detection with the retrained rained detector was the "Pedestrian" class. We believe that two factors contributed to this limitation:

1. The size of the objects representing the "Pedestrian" class was mostly smaller than the other two class objects. This meant that these objects were more susceptible to the presence of rain, which usually occluded and distorted all or most of the pixels representing this class in the image.
2. The number of occurrences of the "Pedestrian" object in the dataset we used for training was much smaller than the other two. We counted 15 "Pedestrian" object instances in the whole training dataset, compared to the thousands of occurrences for the other two classes. Our dataset was collected on motorways in Michigan and under rainy conditions, so the presence of pedestrians was the exception rather than the norm.

We also verified that the retrained detector performance did not degrade under rain-free conditions by retesting the rained detector with the original rain-free dataset. The retrained rained model performance was similar to that of the rain-free detector under the rain-free dataset which made us conclude that the retrained detector retained the information learned by the original rain-free-trained detector model.

The semantic segmentation test cases provided similar results to the object detection ones. The rain-free segmentation model performed well under rain-free conditions, but its performance degraded when tested with rained image dataset. The degradation level was not as severe as that observed in the object detection application. This can be partially attributed to the fact that in the segmentation model, the classes were much larger than those in the object detection application, and thus less susceptible to the presence of raindrops in the input images.

The segmentation model trained on the rain-free dataset performed worse on the de-rained images dataset than on the original rained images dataset. The performance of the retrained image segmentation model showed considerable improvement in segmenting rained images after the baseline rain-free model was trained with the simulated raindrops dataset. Retesting the retrained image segmentation model with the rain-free dataset showed a performance improvement over the rain-free model. The performance gain can be partially attributed to retraining the rain-free model with simulated rained images that were based on the rain-free ones. We argue that the rained images acted as a transformed version of the original ones, even if the transformation caused some level of image quality degradation. In that sense, the rained images augmented the original rain-free dataset, and image augmentation is a standard technique used in the training of the DNN to improve performance. One may argue that the retraining process tuned the detection and segmentation model parameters, which allowed them to denoise

(de-rain) the input images, before being used for the detection and segmentation tasks. We present the following evidence to refute that argument.

1. The data sets used for retraining did not include any information about the raindrops (raindrop labels) to help the system identify them and "work around them" for the detection and segmentation tasks. Rather than raindrops, the labeled objects of interest in both models were commonly encountered in a drive cycle, including cars, trucks, sky, road, and pedestrians.
2. Only rained datasets were used for retraining, not matched pairs of clear and rained images. so even if the detection and segmentation networks could learn raindrops, they were not fed with datasets to facilitate this presumed learning capability.
3. The DNN architectures we built our detection and Segmentation models on were not of the GAN "family" so it was unlikely that they could discriminate raindrops and eliminating them through training.
4. Training a DNN designed for the de-raining task takes a lot more training time and a bigger training set than what we used in our detection and segmentation models. The default setting for epochs in the implementation of Quin et al. algorithm was set to 4000 [11]. The implementers of Quan et al. reported needing one hundred thousand epochs of training to achieve their results [76]. We, in comparison needed around 300 epochs to train either one of our models to a good level of performance.
5. The performance improvement was observed on both models that were developed with different architectures. If we could arguably assume that the DNN architecture of one model allowed for raindrop detection and removal, it is less likely that another model's DNN achieved the same deed.

As we mentioned earlier, the experiments involving using de-rained datasets that were generated by a state-of-the-art de-raining algorithm showed worse performance than the ones using rained datasets without de-raining. The same findings were reported by Li et al. [90] and another one on haze removal done by Pei et al. [91]. "since those deraining algorithms were not trained/optimized towards the end goal of object detection, they are unnecessary to help this goal, and the deraining process itself might have lost discriminative, semantically meaningful true information" [91]. We believe that there may be no de-raining add-on fix to this problem, in a sense that a general-purpose de-raining algorithm can be plugged into the specific vision system (e.g., traffic light recognition) that would improve the system performance under rainy conditions. We believe it is possible to have a hybrid solution, employing both de-raining (denoising) of input images and reducing system sensitivity to raindrops through re-learning and transfer learning (system desensitization). This hybrid approach, however, requires further examination of what important features are removed in the de-raining process, that a given vision application is looking for, to perform its intended functionality.

7.6 Conclusion

We started with a hypothesis that it would be possible to improve the performance of vision algorithms developed as DNN models under rained conditions, by retraining these models with rained image samples. We also hypothesized that the performance improvements could be in-bar with the improvements achieved by de-raining the input images. To put it in more generic terms, we proposed that decreasing the system's sensitivity to noise could provide similar levels of overall system performance as those achieved by denoising the noisy system inputs.

To verify our hypothesis, we selected two common vision applications, namely object detection, and semantic image segmentation, and developed and trained their models using different rain-free and rained image datasets. The results proved that retraining improves vision system performance under rainy conditions, actively expanding the useful application domain to include both clear and rained conditions.

Unexpectedly, the de-raining process degraded both system performances even more than under rained image dataset as the system input. The state-of-the-art de-raining algorithms aim to optimize the de-rained output image in terms of similarity to a clear one. It might be that, though visually similar to a clear image, the de-rained image has some features masked or distorted through the de-raining process, that otherwise would have been used by a target vision application (e.g., objection detection) to learn desirable characteristics of the image and its components. We conclude that due to the lack of a generic de-raining module that can be plugged in before any vision algorithm, retraining with rained images is the better solution to the problem.

We also showed the benefit of using synthetic data sets, in the form of images with generated raindrops, in retraining both object detection and image segmentation application models. Labels objects or segments in the original (clear) images can be reused with the synthetic dataset unchanged. Both the object detector and semantic segmentation models were successfully trained with synthetic datasets and, when tested with real rain datasets, showed great performance improvements. In addition, the synthetic dataset seems to play a secondary role as an image augmenter to the rain-free dataset which contributes to further performance improvements under clear rain conditions.

Chapter 8

Conclusions

The automotive domain is a highly regulated domain with a strong focus on safety and robustness under all driving conditions. Self-driving vehicles and autonomous driving have gained lots of momentum in the last few years, and different vehicle OEMs are already producing vehicles with some level of self-driving capabilities. Vision-based systems play an integral role in many of these self-driving vehicles. The recent improvements of deep-learning vision algorithms further increased the dependency on vision-based systems to the extent that Tesla has recently announced that its new models will drop in-vehicle RADAR systems for autonomous driving, and solely rely on vision-based systems [92].

It is imperative to have a clear understanding of the limitations of vision-based systems, performing under different conditions, including adverse weather conditions. This dissertation presents a comprehensive study of rain as a type of adverse weather condition and its effect on the degradation of input images and vision-based algorithms. It also describes some novel techniques for simulating rain presence in input images, and techniques to improve the performance of vision-based algorithms under rainy conditions.

8.1 Dissertation Summary

Rain in input images can be present in two main forms, as falling rain streaks or as adherent raindrops on lens covers or vehicle windshields. In the automotive domain, falling rain is the

main source of degradation of input images for infrastructure applications, such as automatic traffic lights control and traffic monitoring. The cameras for these applications are usually covered in a way that protects the lens and lens covers from direct contact with rain. We developed a falling rain simulation model that randomly generates rain streaks and adds them to rain-free source images. the output is a stereo pair of color images, with rain streaks varying in size, density, and brightness, based on local and global characteristics of the source image scene, and those of the rain streaks. We also developed a falling rain de-raining algorithm, based on some selected properties of rain streaks in the image. We used the generated images from our falling rain simulator to test the performance of the de-raining algorithm.

In the case of ego moving vehicle with an in-vehicle camera, and in some surveillance applications, adherent raindrops to vehicle windshields and lens protective covers becomes the dominant source of degradation to input images of vision-based applications. we developed a novel adherent raindrop simulator model, that takes rain-free images and adds realistic adherent raindrops to them. The result is a color, rained image, with adherent raindrops of different sizes, shapes, and positions, all controllably randomized. Unlike the main-stream ray-tracing approach for simulating adherent raindrops, we employed the raindrop lens barrel effect and image transformation techniques to generate realistic adherent raindrops in images.

We studied the effect of the image degradation caused by the adherent raindrops on the performance of some vision-based algorithms. Although there is a general acceptance in the research body of the detrimental effect of degraded images on the overall performance of vision-based algorithms, there has not been a complete quantitative study of that effect, to the best of our knowledge. We measured the performance of some state-of-the-art object detection algorithms on clear and rained image sets, and showed a clear correlation between image quality

and the performance of these object detectors, both in terms of the percentage of correct detections, and the confidence levels of detection of the identified objects.

One observation from this quantitative analysis was that the object detectors under test showed some level of robustness to raindrop presence in the input images. This made us wonder if we can further improve this robustness, by further decreasing the sensitivity of the vision-based algorithms to the image degradation due to raindrops. We hypothesized that by applying transfer-learning and relearning with rained images, we could achieve a performance boost of the vision-based algorithms that were in-bar with using a de-raining stage to preprocess the input images. We tested our hypothesis on state-of-the-art object detection and semantic segmentation models. The models were first trained with rain-free image sets, then retrained with rained images that we generated from our adherent raindrop simulation model. We tested the performance of the two algorithm models on real rain datasets that we had previously collected from real drive cycles and showed clear performance boosts in both. The de-rained images did not provide a similar performance boost but rather showed lower performance compared to using the original rained images as input to the two models. We will describe this drawback further in the section.

8.2 Limitations and Future Work

This section describes some limitations we encountered in our work and some ideas for future improvements.

8.2.1 Falling Rain Streaks Simulator

Our simulation model produced images with rain streaks that resemble real falling rain from the perspective of a human observer. We could not however quantitatively test these results since we lacked a good dataset of real falling rain images. to the extent of our knowledge, there is no

publicly available dataset of images with falling rain streaks, that is captured with dedicated outdoor cameras. It is beneficial to create such a dataset, with cameras of a type similar to the ones used for traffic control or automatic traffic light activation.

8.2.2 Falling Rain Detection and Removal

We developed an algorithm for falling rain streaks detection that was based on the physical and brightness properties of the rain streaks. This technique worked well in most cases but there were still instances where irregular rain streaks (per our constraints) were not detected. Adding chromatic properties may improve the detection rate of rain streaks. Deep-learning algorithms showed good results in restoring images with rain streaks to their rain-free versions. As we described in this dissertation, these algorithms were trained on synthesized rain streaks and it would be interesting to see how well they perform on real falling rain datasets.

Moving objects in the image background introduced lots of noise in the falling rain streaks removal process. Scene flow information may improve image restoration if it is integrated into the falling rain streak detection process.

8.2.3 Adherent Raindrop Simulator

Our simulator produced images with realistic raindrops that were generated by employing optical properties of the raindrops (the fisheye lens effect), and intensity and chromatic properties. We developed our model under the Mathworks suite, using functions from the image processing toolset in MATLAB 2021a. The algorithm was not optimized for speed and it would be interesting to rewrite some time-consuming functions using more efficient languages (e.g., C++) or even rewriting the whole algorithm in a language with built-in image-processing capabilities, such as Python.

8.2.4 Using Relearning to Improve Performance of DNN-Based Vision Algorithms

Our research showed clear improvement in the performance of algorithms for object detection and semantic segmentation. The de-raining approach, on the other hand, degraded the performance even more than using rained images on detection and segmentation models that were not retrained with rained images. The de-raining algorithm may have removed some important features from the input images that otherwise would have helped in the detection of objects and segmentation of image scenes. To test this hypothesis, we can examine the DNN features learned in the final layers of the DNNs, closest to the output layer. By comparing the type of features learned with rained, de-rained, and rain-free image sets, we may be able to determine which features were omitted or modified by the de-raining process. This study, if successful, may help design better de-raining algorithms that preserve the features critical for the vision-based algorithm under development.

References

[1] H. Kong, H. C. Akakin and S. E. Sarma, "A Generalized Laplacian of Gaussian Filter for Blob Detection and Its Applications," *IEEE Transactions on Cybernetics,* vol. 43, no. 6, pp. 1719-1733, 12 2013.

[2] D. Marr and E. Hildreth, "Theory of Edge Detection," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 207, no. 1167, pp. 187-217, 1980.

[3] G. Wang, C. Lopez-Molina and B. D. Baets, "Automated blob detection using iterative Laplacian of Gaussian filtering and unilateral second-order Gaussian kernels," *Digital Signal Processing*, vol. 96, p. 102592, 2020.

[4] C. Godard, O. M. Aodha and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition , 2017.

[5] D. C. BROWN and E. Gallie, "Decentering Distortion of Lenses.," *Photogrammetric Engineering*, vol. 32, no. 3, pp. 444-462, 1966.

[6] R. Vreja and R. Brad, "Image Inpainting Methods Evaluation and Improvement," *The Scientific World Journal*, vol. 2014, no. https://doi.org/10.1155/2014/937845, 2014.

[7] Y. Liu and C. Shu, "A comparison of image inpainting techniques," in Sixth International Conference on Graphic and Image Processing (ICGIP 2014), Beijing, China, 2015.

[8] S. Shivaranjani and R. Priyadharsini, "A Survey on Inpainting Techniques," in International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 2016.

[9] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600 - 612, 4 APRIL 2004.

[10] MathWorks, "Image Quality Metrics," MathWorks, Inc., 1 1 2021. [Online]. Available: https://www.mathworks.com/help/images/image-quality-metrics.html. [Accessed 8 5 2021].

[11] R. Qian, R. T. Tan, W. Yang, J. Su and J. Liu, "Attentive Generative Adversarial Network for Raindrop Removal from A Single Image," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, Utah, 2018.

[12] Wikipedia contributors, "Peak signal-to-noise ratio," 14 4 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Peak_signal-to-noise_ratio&oldid=887764757. [Accessed 26 5 2019].

[13] N. Tyagi, "L2 and L1 Regularization in Machine Learning," Analytic Steps, 28 2 2021. [Online]. Available: https://www.analyticssteps.com/blogs/l2-and-l1-regularization-machine-learning. [Accessed 9 5 2021].

[14] D. Padfield, "Generalized Normalized Cross Correlation," MathWorks, 19 4 2012. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/29005-generalized-normalized-cross-correlation. [Accessed 1 10 2019].

[15] Y. Rubner, C. Tomasi and L. J. Guibas, "A metric for distributions with applications to image databases," in Sixth International Conference on Computer Vision, Bombay, India, 1998.

[16] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation," School of Informatics and Engineering, Flinders University of South Australia, Adelaide, 2007.

[17] Wikipedia contributors, "Jaccard index," 1 4 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Jaccard_index&oldid=890384326. [Accessed 26 5 2019].

[18] Wikipedia contributors, "Sørensen–Dice coefficient," 9 4 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=S%C3%B8rensen%E2%80%93Dice_coefficient&oldid=891683118. [Accessed 26 5 2019].

[19] W. Mantzel, "F1/Dice-Score vs IoU," 13 11 2017. [Online]. Available: https://stats.stackexchange.com/q/276144. [Accessed 26 5 2019].

[20] M. Everingham and J. Winn, "The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Development Kit," in Visual Object Classes Challenge 2011 (VOC2011), 2011.

[21] "Evaluate," COCO Consortium , 29 6 2021. [Online]. Available: https://cocodataset.org/#detection-eval. [Accessed 8 8 2021].

[22] K. Garg and S. K. Nayar, "When Does a Camera See Rain?," in Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05), New York, New York, 2005.

[23] W. C. Chia, L. S. Yeong and S. I. Ch'ng, "The effect of rainfall on feature points extraction and image stitching," in International Conference on Information Science, Electronics and Electrical Engineering, Sapporo, Japan, 2014.

[24] Y. Hamzeh, Z. El-Shair and S. A. Rawashdeh, "Effect of Adherent Rain on Vision-Based Object Detection Algorithms," *SAE International Journal of Advances and Current Practices in Mobility*, vol. 2, pp. 3051-3059, 2020.

[25] Y. Hamzeh and S. A. Rawashdeh, "A Review of Detection and Removal of Raindrops in Automotive Vision Systems," *Journal of Imaging*, vol. 7, no. 3, p. 52, 2021.

[26] A. Cord and N. Gimonet, "Detecting Unfocused Raindrops In-Vehicle Multipurpose Cameras," *IEEE Robotics & Automation Magazine*, vol. 21, no. 1, pp. 49-56, 2014.

[27] H. Kurihata, T. Takahashi, k. Ide, Y. Mekada, H. Murase, Y. Tamatsu and T. Miyahara, "Rainy Weather Recognition from In-Vehicle Camera Images for Driver Assistance," in IEEE Proceedings. Intelligent Vehicles Symposium, Las Vegas, NV, USA, USA, 2005.

[28] E. Fouad, E. Abdelhak and A. Salma, "Modelisation of raindrops based on declivity principle," in 13th International Conference on Computer Graphics, Imaging and Visualization (CGiV), Beni Mellal, Morocco, 2016.

[29] J. C. Halimeh and M. Roser, "Raindrop Detection on Car Windshields Using Geometric-Photometric Environment Construction and Intensity-Based Correlation," in 2009 IEEE Intelligent Vehicles Symposium (IV), Xi'an, Shaanxi, China, 2009.

[30] M. Roser and A. Geiger, "Video-based raindrop detection for improved image registration," in 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, Kyoto, Japan, 2009.

[31] M. Sugimoto, N. Kakiuchi, N. Ozaki and R. Sugawara, "A Novel Technique for Raindrop Detection on a Car Windshield using Geometric-Photometric Model," in 15th International IEEE Conference on Intelligent Transportation Systems, Anchorage, Alaska, USA, 2012.

[32] I. Stuppacher and P. Supan, "Rendering of Water Drops in Real-Time," in Central European Seminar on Computer Graphics for Students, 2007.

[33] M. Roser, J. Kurz and A. Geiger, "Realistic Modeling of Water Droplets for Monocular Adherent Raindrop Recognition using Bézier Curves," in Asian Conference on Computer Vision, Springer, Berlin, Heidelberg, 2010.

[34] X. Yan, Y. Luo and X. Zheng, "Weather Recognition Based on Images Captured by Vision System in Vehicle," in Advances in Neural Networks - 6th International Symposium on Neural Networks (ISNN 2009), Wuhan, China, 2009.

[35] Q. Wu, W. Zhang and B. V. Kumar, "RAINDROP DETECTION AND REMOVAL USING SALIENT VISUAL FEATURES," in 2012 19th IEEE International Conference on Image Processing, Orlando, FL, USA, 2012.

[36] H.-C. Liao, J. Shin, D.-Y. Wang and C.-L. Yang, "Video-based Water Drop Detection and Removal Method for a Moving Vehicle," *Information Technology Journal,* vol. 12, no. 4, pp. 569-583, 2013.

[37] J. Ishizuka and K. Onoguchi, "Detection of Raindrop with Various Shapes on a Windshield," in 5th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2016),, Roma, Italy , 2016.

[38] A. Yamashita, M. Kuramoto, T. Kaneko and K. T. Miura, "A Virtual Wiper – Restoration of Deteriorated Images by Using Multiple Cameras –," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), Las Vegas, NV, USA, 2003.

[39] A. Yamashita, T. Kaneko and K. T. Miura, "A Virtual Wiper –Restoration of Deteriorated Images by Using a Pan-Tilt Camera-," in IEEE International Conference on Robotics and Automation, New Orleans, LA, USA,, 2004.

[40] A. Yamashita, I. Fukuchi, T. Kaneko and K. T. Miura, "Removal of Adherent Noises from Image Sequences by Spatio-Temporal Image Processing," in IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 2008.

[41] A. Yamashita, Y. Tanaka and T. Kaneko, "Removal of Adherent Waterdrops from Images Acquired with Stereo Camera," in IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Alta., Canada, 2005.

[42] M. Roser and F. Moosmann, "Classification of Weather Situations on Single Color Images," in IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, 2008.

[43] A. Cord and D. Aubert, "Towards Rain Detection through use of In-Vehicle Multipurpose Cameras," in 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, 2011.

[44] F. Nashashibi, R. D. Charette and A. Lia, "Detection of Unfocused Raindrops on a Windscreen using Low Level Image Processing," in International Conference on Control, Automation, Robotics and Vision : ICARV'2010, Singapour, Singapore, 2010.

[45] D. Eigen, D. Krishnan and R. Fergus, "Restoring An Image Taken Through a Window Covered with Dirt or Rain," in 2013 IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 2013.

[46] Y. Quan, S. Deng, Y. Chen and H. Ji, "Deep Learning for Seeing Through Window With Raindrops," in IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019.

[47] Z. Hao, S. You, Y. Li, K. Li and F. Lu, "Learning From Synthetic Photorealistic Raindrop for Single Image Raindrop Removal," in IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea (South), 2019.

[48] D. P. Kingma and J. L. Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION," in The 3rd International Conference for Learning Representations, San Diego, 2015.

[49] J. Kim, D. Huh, T. Kim, J. Kim, J. Yoo and J.-S. Shim, "Raindrop-Aware GAN: Unsupervised Learning for Raindrop-Contaminated Coastal Video Enhancement," *Remote Sensing,* vol. 12, no. 20, p. 3461, 2020.

[50] A. Mittal, R. Soundararajan and A. C. Bovik, "Making a "Completely Blind" Image Quality Analyzer," *IEEE Signal Processing Letters*, pp. 209-212, 3 2013.

[51] S. Alletto, C. Carlin, L. Rigazio, Y. Ishii and Tsukizawa, "Adherent Raindrop Removal with Self-Supervised Attention Maps and Spatio-Temporal Generative Adversarial Networks," in IEEE/CVF International Conference on Computer Vision Workshops, Seoul, Korea, 2019.

[52] A. Palazzi, D. Abati, S. Calderara, F. Solera and R. Cucchiara, "Predicting the Driver's Focus of Attention: The DR(eye)VE Project," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1720-1733, 2019.

[53] J. Peng, Y. Xu, T. Chen and Y. Huang, "Single-image raindrop removal using concurrent channel-spatial attention and long-short skip connections," *Pattern Recognition Letters*, vol. 131, pp. 121-127, 2020.

[54] D. Ren, J. Li, M. Han and M. Shu, "Not All Areas Are Equal: A Novel Separation-Restoration-Fusion Network for Image Raindrop Removal," in *Pacific Graphics*, Wellington, New Zealand, 2020.

[55] M. Yang, K. Yu, C. Zhang, Z. Li and K. Yang, "Denseaspp for semantic segmentation in street scenes," in IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018.

[56] P. Barnum, T. Kanade and S. G. Narasimhan, "Spatio-Temporal Frequency Analysis for Removing Rain and Snow from Videos," in IEEE Workshop on Photometric Analysis For Computer Vision (PACV), in conjunction with ICCV, 2007.

[57] Y. Hamzeh and S. Rawashdeh, "Framework for simulating and removing rain in stereo-image videos," in IEEE International Conference on Electro/Information Technology (EIT), Oakland, Michigan, 2018.

[58] S. Starik and M. Werman, "Simulation of Rain in Videos," in Texture Workshop, ICCV, 2003.

[59] K. Garg and S. K. Nayar, "Detection and Removal of Rain from Videos," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04), Washington, DC, USA, 2004.

[60] STEREOLABS, "Introducing the ZED Mini," STEREOLABS, 2018. [Online]. Available: https://www.stereolabs.com/. [Accessed 15 2 2018].

[61] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in Conference on Computer Vision and Pattern Recognition (CVPR)}, Rhode Island, 2012.

[62] K. GARG and S. K. NAYAR, "Vision and Rain," *International Journal of Computer Vision*, vol. 75, no. 1, pp. 3-27, 2007.

[63] Z. Shi, J. Long, W. Tang and C. Zhang, "Single image dehazing in inhomogeneous atmosphere," *Optik-international journal for light and electron optics*, vol. 125, no. 15, pp. 3868-3875, 2014.

[64] Q. Zhu, J. Mai and L. Shao, "Single Image Dehazing Using Color Attenuation Prior," in British Machine Vision Conference (BMVC), Nottingham, UK, 2014.

[65] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single Shot MultiBox Detector," in European conference on computer vision, Springer, Cham, , 2016.

[66] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[67] J. W. Francis, "SSD-VGG-300 Trained on PASCAL VOC Data," The Wolfram Neural Net Repository, 29 10 2018. [Online]. Available: https://github.com/sfzhang15/RefineDet. [Accessed 1 4 2019].

[68] W. N. N. R. implementation, "YOLO V3 Trained on Open Images Data," 30 9 2019. [Online]. Available: https://resources.wolframcloud.com/NeuralNetRepository/resources/YOLO-V3-Trained-on-Open-Images-Data/. [Accessed 15 10 2019].

[69] X. Chen, M. Buckler, J. Serrino, S. T. Cheedella and G. Vasilakis, "tf-faster-rcnn," 2 3 2017. [Online]. Available: https://github.com/endernewton/tf-faster-rcnn. [Accessed 1 9 2019].

[70] Wikipedia contributors, "Mean squared error," 4 5 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Mean_squared_error&oldid=895521601. [Accessed 26 5 2019].

[71] Wikipedia contributors, "Precision and recall," 13 5 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=896900450. [Accessed 25 5 2019].

[72] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in Advances in neural information processing systems, 2015.

[73] MathWorks, "Create a Gallery of Transformed Images," MathWorks Inc., [Online]. Available: https://www.mathworks.com/help/images/creating-a-gallery-of-transformed-images.html. [Accessed 25 3 2021].

[74] S. You, R. T. Tan, R. Kawakami, Y. Mukaigawa and K. Ikeuchi, "Adherent Raindrop Modeling, Detection and Removal in Video," *IEEE Transaction on Pattern Recognition and Machine Intelligence*, vol. 38, no. 9, pp. 1721-1733, 2016.

[75] C. Carlin, "PSVL/OpenSynthrain," 22 6 2020. [Online]. Available: https://github.com/PSVL/OpenSynthrain. [Accessed 1 6 2021].

[76] Y. Quan, S. Deng, Y. Chen and H. Ji, "ljm619/raindropAttention," 7 10 2019. [Online]. Available: https://github.com/ljm619/raindropAttention. [Accessed 1 7 2021].

[77] R. Qian, R. T.Tan, W. Yang, J. Su and J. Liu, "rui1996 / DeRaindrop," GitHub, 29 6 2018. [Online]. Available: https://github.com/rui1996/DeRaindrop. [Accessed 20 1 2021].

[78] R. Yasarla and V. M. Patel, "Uncertainty Guided Multi-Scale Residual Learning-using a Cycle Spinning CNN for Single Image De-Raining," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, California, 2019.

[79] R. Yasarla and V. M. Patel, "rajeevyasarla/UMRL--using-Cycle-Spinning," 5 3 2020. [Online]. Available: https://github.com/rajeevyasarla/UMRL--using-Cycle-Spinning. [Accessed 1 8 2021].

[80] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, Hawaii, 2017.

[81] Y. Hamzeh, Z. A. El-Shair, A. Chehade and S. A. Rawashdeh, "Dynamic Adherent Raindrop Simulator for Automotive Vision Systems," *IEEE Access,* vol. 9, pp. 114808-114820, 2021.

[82] MathWorks, "Object Detection Using YOLO v3 Deep Learning," MathWorks, 2021. [Online]. Available: https://www.mathworks.com/help/vision/ug/object-detection-using-yolo-v3-deep-learning.html. [Accessed 7 7 2021].

[83] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," arXiv:1602.07360v4 [cs.CV] , 2016.

[84] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM,* vol. 60, pp. 84-90, 2012.

[85] P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE transactions on pattern analysis and machine intelligence* , pp. 743-761, 8 2011.

[86] MathWorks, "Semantic Segmentation Using Deep Learning," MathWorks, 2021. [Online]. Available: https://www.mathworks.com/help/vision/ug/semantic-segmentation-using-deep-learning.html. [Accessed 15 7 2021].

[87] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff and H. Adam, "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation," in Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 2018.

[88] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in arXiv:1512.03385v1 [cs.CV] , 2015.

[89] MathWorks, "evaluateSemanticSegmentation," 2021. [Online]. Available: https://www.mathworks.com/help/vision/ref/evaluatesemanticsegmentation.html. [Accessed 1 8 2021].

[90] S. Li, I. B. Araujo, W. Ren, Z. Wang, E. K. Tokuda2, R. H. Junior2, R. Cesar-Junior, J. Zhang, X. Guo and X. Cao, "Single Image Deraining: A Comprehensive Benchmark Analysis," *Computer Vision and Pattern Recognition* , no. arXiv:1903.08558v1 [cs.CV]), 2019.

[91] Y. Pei, Y. Huang, Q. Zou, Y. Lu and S. Wang, "Does Haze Removal Help CNN-based Image Classification?," in Proceedings of the European Conference on Computer Vision (ECCV). , 2018.

[92] Tesla, "Transitioning to Tesla Vision," Tesla, 5 2021. [Online]. Available: https://www.tesla.com/support/transitioning-tesla-vision. [Accessed 12 6 2021].

[93] U. Yilmaz, "The Earth Mover's Distance," 12 2 2009. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/22962-the-earth-mover-s-distance. [Accessed 2 11 2020].

[94] Y. RUBNER, C. TOMASI and L. J. GUIBAS, "The Earth Mover's Distance as a Metric for Image Retrieval," *International Journal of Computer Vision,* vol. 40*,* no. 2, pp. 99-121, 2000.

[95] J. Matas, O. Chum, M. Urban and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and vision computing, vol. 22*, no. 10, pp. 761-767, 2004.

[96] F. Bukhari and M. N. Dailey, "Automatic Radial Distortion Estimation from a Single Image," *Journal of Mathematical Imaging and Vision* , vol. 45, no. https://doi.org/10.1007/s10851-012-0342-2, pp. 31-45, 2013.

[97] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[98] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," Advances in Neural Information Processing Systems (NIPS 2017), vol. 30, no. arXiv:1706.08500v6 [cs.LG] , 2018.

[99] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz and B. Catanzaro, "Video-to-Video Synthesis," in Conference on Neural Information Processing Systems (NeurIPS), Montreal, Canada , 2018.

[100] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia and R. Jozefowicz, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," arXiv:1603.04467, pp. 1-19, 2016.

[101] J. Dai, Y. Li, K. He and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," arXiv:1605.06409v2 [cs.CV] , 2016.

[102] D. Nguyen, K. Nguyen, S. Sridharan, I. Abbasnejad, D. Dean and C. Fookes, "Meta Transfer Learning for Facial Emotion Recognition," in 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 2018.

[103] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel and D. Wierstra, "PathNet: Evolution Channels Gradient Descent in Super Neural Networks," arXiv:1701.08734v1 [cs.NE] , 2017.

[104] S. Haq and P. J. Jackson, "Speaker-Dependent Audio-Visual Emotion Recognition," in Auditory-Visual Speech Processing (AVSP) , Norwich, UK, 2009.

[105] O. Martin, I. Kotsia, B. Macq and I. Pitas, "The enterface' 05 audiovisual emotion database," in 22nd International Conference on Data Engineering Workshops (ICDEW'06), 2006.

[106] J. Talukdar, S. Gupta, P. S. Rajpura and R. S. Hegde, "Transfer Learning for Object Detection using State-of-the-Art Deep Neural Networks," in 5th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, Delhi-NCR, 2018.