Depth Sensing

Abishek M [CB.EN.U4CSE20601].[1] ,Shankara Narayanan V [CB.EN.U4CSE20656].[2*] , Sneha Varsha M [CB.EN.U4CSE20659].[3*], Syed Ashfaq Ahmed [CB.EN.U4CSE20665][4*], Tarun Ramaswamy [CB.EN.U4CSE20666][5*]*

[1,2,3,4,5]Amrita School of Engineering, Amrita Vishwa Vidyapeetham
19CSE 431 Digital Image Processing

***Abstract*- Images taken from various locations and situations are used to run filters on in order to preprocess them for depth sensing.**

Keywords-Images, filters, preprocessing, depth sensing

I. Introduction

The motive of this project is to use images taken from a dataset available on Kaggle to run through some preprocessing techniques in both spatial and frequency domain for the process of depth sensing. Edge Detection is carried out as one of the first steps towards depth sensing.

1. Basic Preprocessing

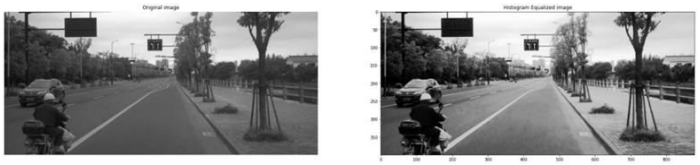
1.1 Histogram Equalization

Histogram equalization is a technique used in image processing to adjust the contrast of an image by spreading out the intensity values of the pixels. It works by transforming the intensity values of the pixels in the image so that the histogram of the image is flattened.

The histogram of an image is a graph that shows the number of pixels in the image at each intensity value. An image with a narrow range of intensity values will have a histogram that is peaked, while an image with a wide range of intensity values will have a flatter histogram.

Histogram equalization works by transforming the intensity values of the pixels in the image so that the histogram becomes more evenly distributed. This can help to improve the contrast of the image and make it easier to see details in both the dark and light areas of the image.

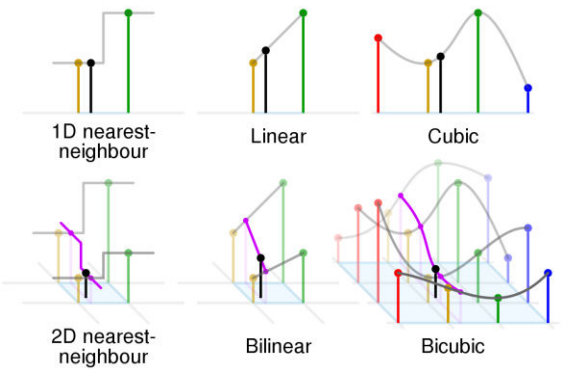
From Left (a),(b): (a) Input Image (b)Histogram Equalized Image



From Left (a),(b): (a) Input Image (b)Histogram Equalized Image

1.2 Interpolation

Interpolation refers to the process of estimating the values of pixels in an image based on the values of surrounding pixels. This can be used to increase the resolution of an image, to change the aspect ratio of an image, or to correct for distortion. There are several different methods that can be used for interpolation, including nearest neighbor interpolation, bilinear interpolation, and bicubic interpolation. The choice of interpolation method will depend on the specific requirements of the application and the trade-off between accuracy and computational complexity.

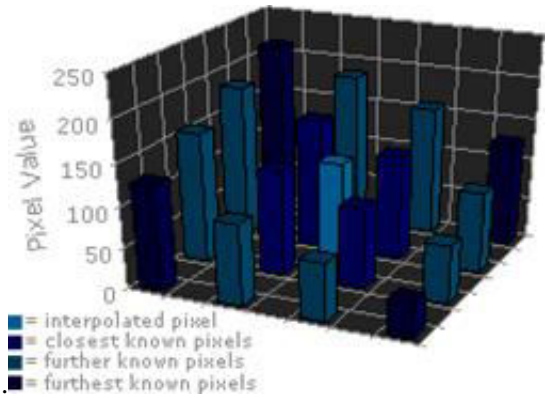


(a) Comparison of graphics of various interpolation techniques

1.3 Cubic Interpolation

Cubic interpolation is a method for image interpolation that estimates the value of a pixel based on a weighted average of the surrounding pixels in the original image. It works by dividing the region around the pixel of interest into a grid of 16 pixels, and using the values of these pixels to estimate the value of the pixel. The weights for the average are calculated based on the distance of the pixel of interest from each of the surrounding pixels, using a cubic function.

Cubic interpolation is more accurate and produces higher-quality images than linear interpolation methods such as bilinear or nearest neighbor interpolation. It is often used in image resizing, image rotation, and other image processing operations that require high-quality contrast or to correct for non-uniform illumination. This type of transformation is based on the logarithmic function, which is a mathematical function that is used to transform data in order to make it more easily interpretable or to bring out certain features that may not be immediately apparent in the original data.



results..

$$p(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$$

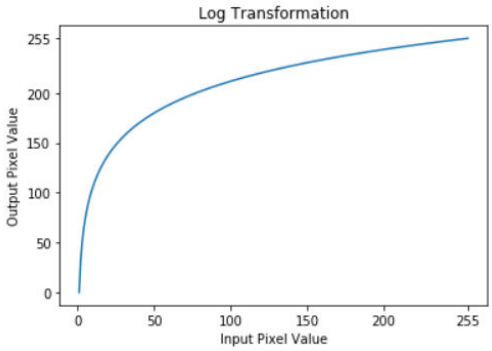
$$p(x,y) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ y \\ y^2 \\ y^3 \end{bmatrix}$$

(a) 3D representation of cubic interpolation (b)Basic Bicubic Interpolation algorithm (c)Expanded algorithm of (a)

(a) Source Image (b) Downscaled image using bicubic interpolation

1.4 Log Transform

In image processing, a logarithmic transformation is often used to stretch the intensity values of an image in order to bring out detail in the shadows or highlight areas of an image that may be too dark or too light. This is achieved by applying the logarithmic function to the pixel values of the image, which has the effect of compressing the values at one end of the intensity range and expanding them at the other end.



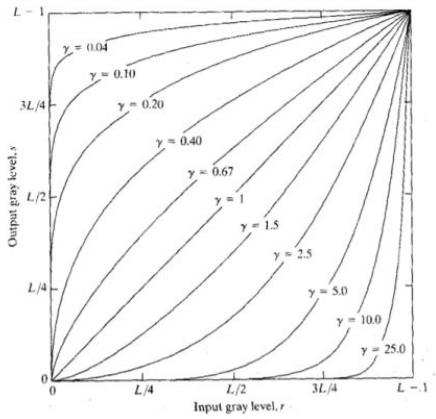
(a) Gray Levels of a log transformed image with respect to the input image

(a):Input Images (b) Corresponding Log Transform

1.5 Gamma Correction

Is a method used in image processing to adjust the luminance of an image. It is often used to correct for the non-linear response of a display device, such as a computer monitor or television, to the intensity of the input signal.

Gamma correction is based on the concept of gamma, which is a measure of the non-linear relationship between the input signal and the output luminance of a display device. Different display devices have different gamma values, which can affect the way that an image is displayed.To correct for these differences in gamma, the input signal is adjusted using a gamma correction curve. This curve maps the input signal to the desired output luminance in a way that is designed to produce a more accurate and pleasing image.



(a) Gray Levels of a gamma corrected image with respect to the input image for different values of gamma

$$S = C \cdot R^{\gamma}$$

(a) Formula used for Gamma Correction of an image

Input Image

(a) Images after Gamma Correction with gamma values of 0.1, 0.5, 1.5, 2.0, 2.5 and 3.0, respectively

Input Image

(a) Images after Gamma Correction with gamma values of 0.1, 0.5, 1.5, 2.0, 2.5 and 3.0 respectively

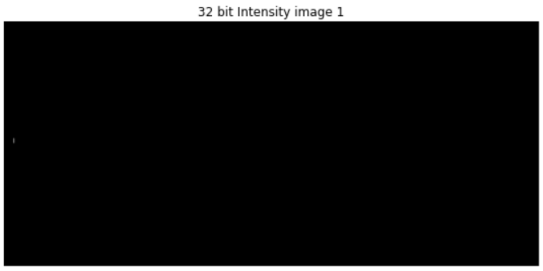
1.6 Intensity Resolution

Intensity resolution, also known as gray level resolution or bit depth, is an important aspect of image processing. It refers to the number of distinct intensity values that can be represented in an image. The intensity resolution of an image is determined by the number of bits used to represent each pixel.

In general, images with a higher intensity resolution will appear more realistic and lifelike because they are able to capture and represent a greater range of intensity values. However, increasing the intensity resolution of an image also increases the file size of the image and the amount of storage and processing required to handle it. It is important to balance the need for high intensity resolution with the constraints of available storage and processing resources.

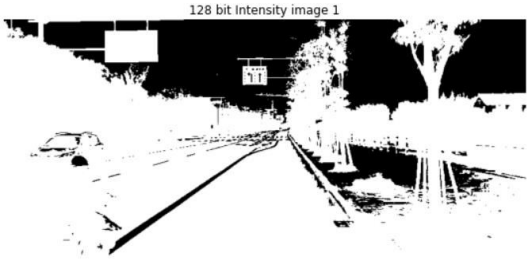
It refers to the number of intensity levels used to represent an image. The more intensity levels used, the finer the level of detail discernible in an image. Intensity level resolution is usually given in terms of the number of bits used to store each intensity level.

For Depth sensing below we used three different intensity levels:- [32 bit, 64, 128].



(a)32 bit Intensity Image

(a)64 bit intensity Image



(a)128 bit Intensity Image

2. Filtering for Noise Reduction

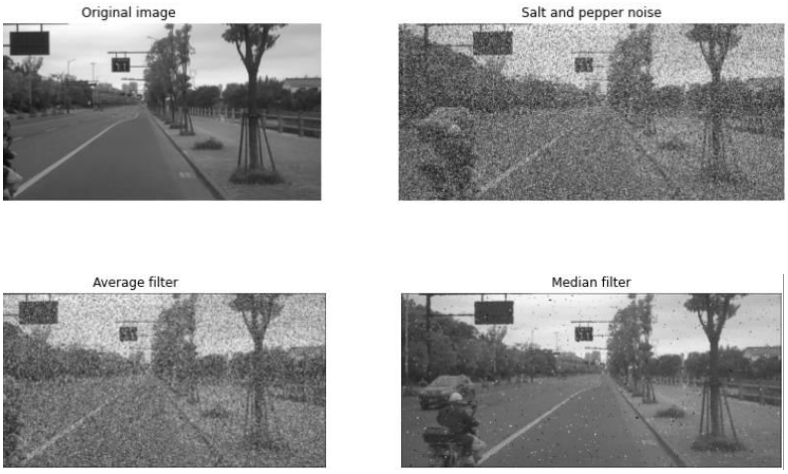
The presence of noise can heavily distort the depth and disparity map calculations. For this, the de noising processes is extremely vital.

2.1 Salt and Pepper Noise Removal

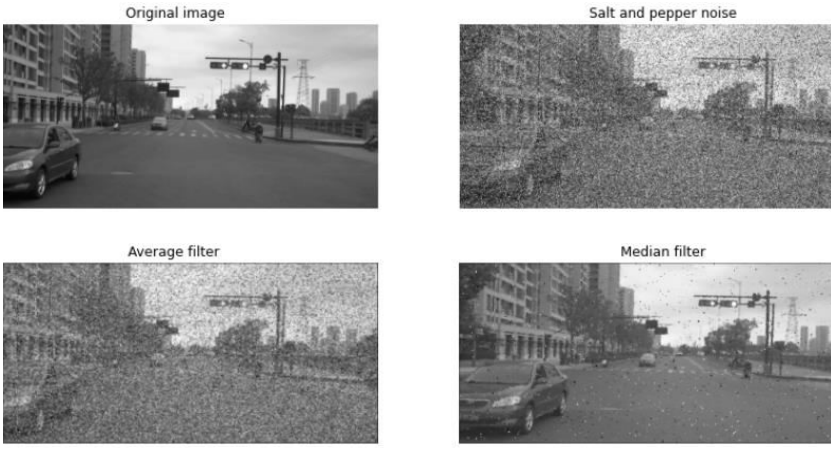
Salt-and-pepper noise is also called impulse noise. It can be caused by several reasons like dead pixels, analog-to-digital conversion error, bit transmission error, etc. The dataset can be cleaned from images containing salt and pepper noise using the following method.

In image processing, salt and pepper noise is a type of noise that is typically characterized by white and black pixels scattered randomly throughout the image. This type of noise is usually caused by electronic interference or other types of damage to the image. It can be difficult to remove salt and pepper noise from an image, because the pixels affected by the noise are randomly distributed and do not follow any particular pattern. One way to remove salt and pepper noise is to use a median filter, which replaces the value of each pixel with the median value of the pixels in its neighborhood. This can help to smooth out the noise and restore the image to its original appearance. Other techniques, such as Gaussian filtering and Wiener filtering, can also be used to reduce the effects of salt and pepper noise.

A median filter is used to clean the images from salt and pepper noise as it is the most efficient among non linear filters. The implementation of both median and average filter is done and plotted side by side for comparison.



(a),(b),(c),(d): Input Image, Salt and Pepper Noise added, Effect of Average Filter,Effect of Median Filter

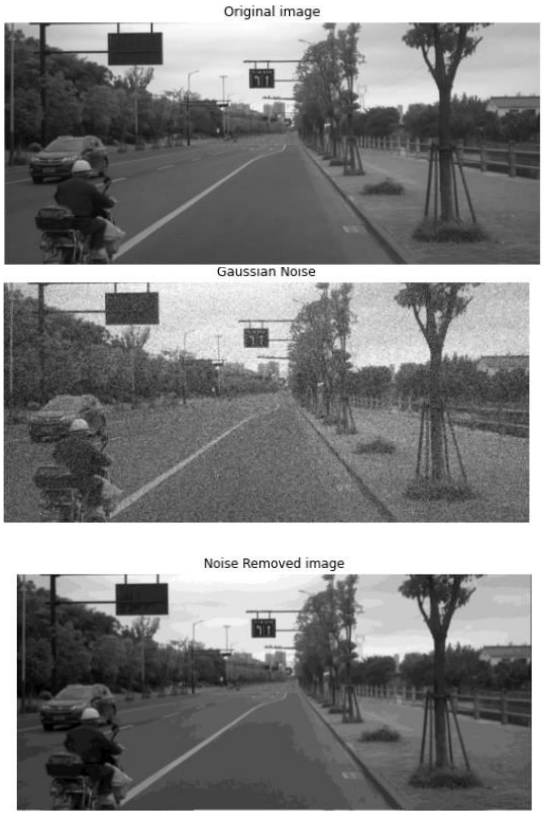


(a),(b),(c),(d): Input Image, Salt and Pepper Noise added, Effect of Average Filter,Effect of Median Filter

2.2 Gaussian Noise Removal

Gaussian noise is a type of noise that is commonly used to add random variations to an image in image processing. It is named after the Gaussian distribution, which is a probability distribution that is characterized by a bell-shaped curve. In image processing, Gaussian noise is typically added to an image to simulate the effects of noise that may be present in real-world images. This can be useful for testing image processing algorithms and for training machine learning models to recognize patterns in noisy images.

The removal of Gaussian noise can be done using any non-linear filters (average, median etc) but Wiener's filter is a type of filter that can do both filtering of Gaussian noise and also fix the motion blur in an image. In the project, there is a high possibility of having images with both Gaussian blur and most importantly motion blur hence correcting them becomes a crucial step towards finding the depth of objects in the image.



(a),(b),(c): Original Image, Image with Gaussian Noise introduced in it, Image after Gaussian noise removal using Wiener filter

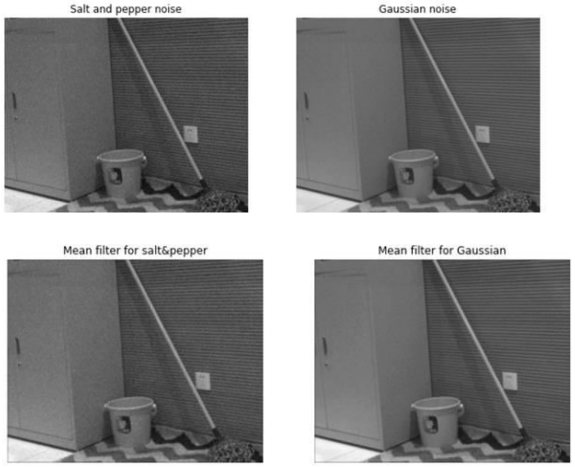
Mean Filters

2.3 Arithmetic Mean Filter

In image processing, an arithmetic mean filter is a simple sliding window spatial filter that replaces the center pixel in the window with the arithmetic mean of all the pixel values in the window. It is often used to smooth images, reduce noise, and reduce detail.

The size of the window, or kernel, determines the extent to which the image is smoothed. A larger kernel size will result in more smoothing, while a smaller kernel size will result in less smoothing.

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

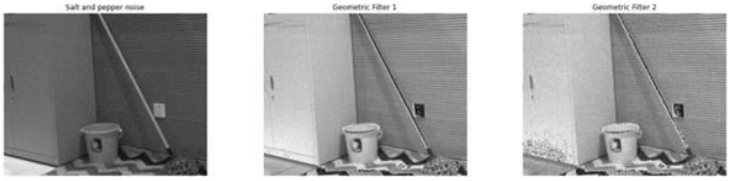


From Top left to bottom right (a):Image with salt and pepper noise (b) Image with Gaussian noise (c)Mean filter for salt and pepper noise (d)Mean filter for Gaussian Noise

2.4 Geometric Mean Filter

In image processing, a geometric mean filter is a spatial filter that replaces the value of each pixel in an image with the geometric mean of the values of the pixels in a local neighborhood around that pixel. The geometric mean is defined as the nth root of the product of the values of the pixels in the neighborhood, where n is the number of pixels in the neighborhood.

$$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$



(a) Image with Salt and Pepper Noise (b) Image filtered with Geometric Filter 1 (c)Image filtered with Geometric Filter 2

2.5 Harmonic Mean Filter

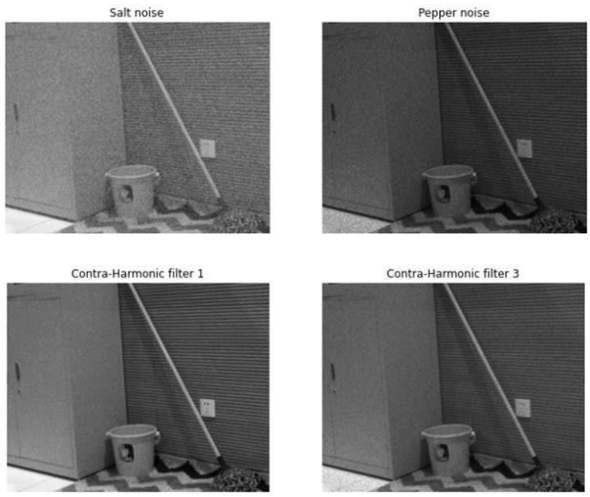
In image processing, a contraharmonic mean filter is a spatial filter that replaces the value of each pixel in an image with the weighted average of the values of the pixels in a local neighborhood around that pixel. The weights are determined by a parameter called the "order" of the filter, which can be either positive or negative.

A contraharmonic mean filter reduces or virtually eliminates the effects of salt-and-pepper noise. For positive values of Q, the filter eliminates pepper noise. For negative values of Q it eliminates salt noise. It cannot do both simultaneously.

Note that the contraharmonic filter is simply the arithmetic mean filter if Q = 0, and the harmonic mean filter if Q = −1.

A larger region (filter size) yields a stronger filter effect with the drawback of some blurring.

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$



(a) Salt Noise (b) Pepper Noise (c)Contra-Harmonic filter with(d)Contra-Harmonic filter

The adaptive median filter works by comparing the intensity of each pixel in the image to the median intensity of the surrounding pixels. If the intensity of the pixel is significantly different from the median intensity, it is considered to be noise and is replaced with the median intensity. This process is repeated for all pixels in the image, resulting in a smoothed and denoised image.

One of the key advantages of the adaptive median filter is its ability to effectively remove salt and pepper noise (impulse noise). It is also effective at preserving edges and other important features in the image, making it a popular choice for image processing applications.

Stage A: $A1 = z_{med} - z_{min}$

$A2 = z_{med} - z_{max}$

If $A1 > 0$ and $A2 < 0$, Go to level B

Else increase the window size

If window size $\leq S_{max}$ repeat level A

Else output z_{med}

Stage B: $B1 = z_{xy} - z_{min}$

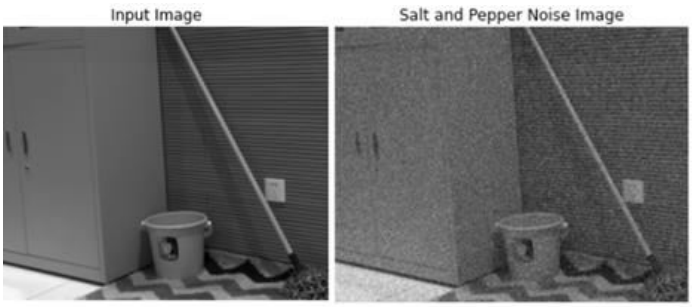
$B2 = z_{xy} - z_{max}$

If $B1 > 0$ and $B2 < 0$, output z_{xy}

Else output z_{med}

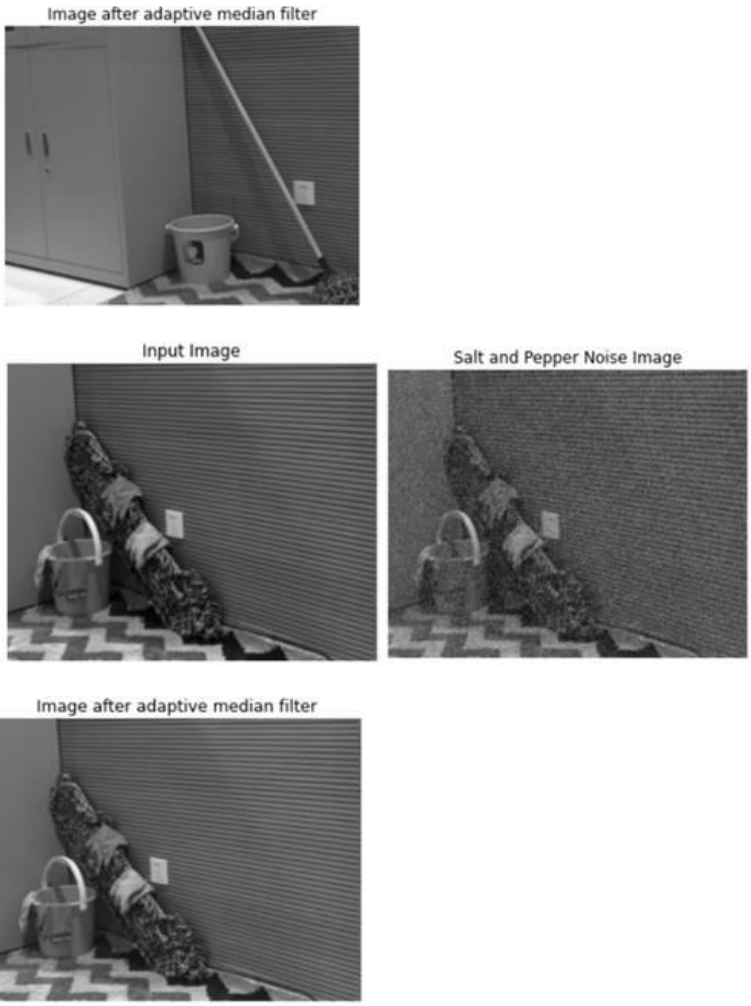
where:

- z_{min} = minimum grey level in S_{xy}
- z_{max} = maximum grey level in S_{xy}
- z_{med} = median of grey levels in S_{xy}
- z_{xy} = grey level at coordinates (x, y)
- S_{max} =maximum allowed size of S_{xy}



2.6 Adaptive Median Filter

The adaptive median filter is a non-linear digital filter used for image processing and computer vision. It is designed to remove noise from images while preserving edges and other important image features.



(a)Input Image (b)Salt and Pepper Noise (c)Image after applying adaptive median filter

2.7 Adaptive Noise Reduction

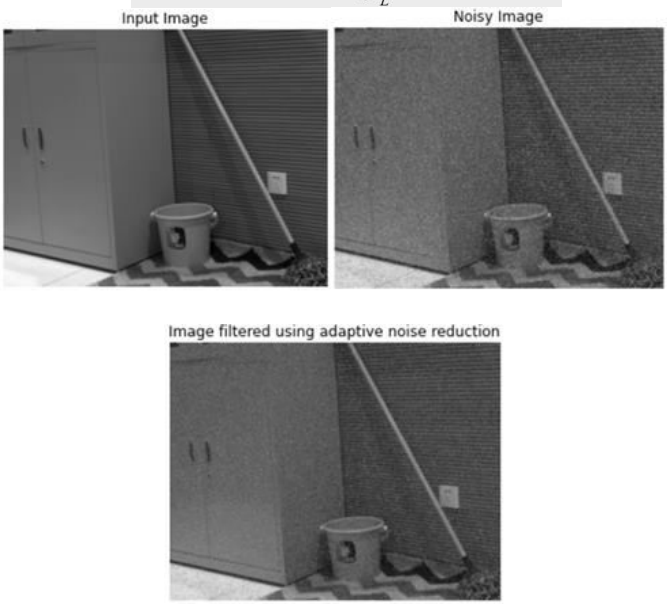
Adaptive noise reduction is a technique used to reduce the level of noise in a signal or image. It involves analyzing the characteristics of the noise in the signal and adapting the noise reduction algorithm to better fit the noise profile.

The technique used here involves using the mean of a neighborhood and variance of both the neighborhood and of the overall image.

One of the key benefits of adaptive noise reduction in image processing is that it can effectively remove noise while preserving edges and other important features in the image. This is important because traditional noise reduction techniques can often blur or distort these features, leading to a loss of detail and clarity in the image.

Adaptive noise reduction is commonly used in a wide range of image processing applications, including medical imaging, surveillance, and industrial inspection. It is also used in digital photography to improve the quality of images captured in low light or high-noise environments.

$$\hat{f}(x,y) = g(x,y) - \frac{\sigma_\eta^2}{\sigma_L^2}[g(x,y) - m_L]$$



(a)Input Image (b)Noisy Image ©Image filtered using adaptive noise reduction

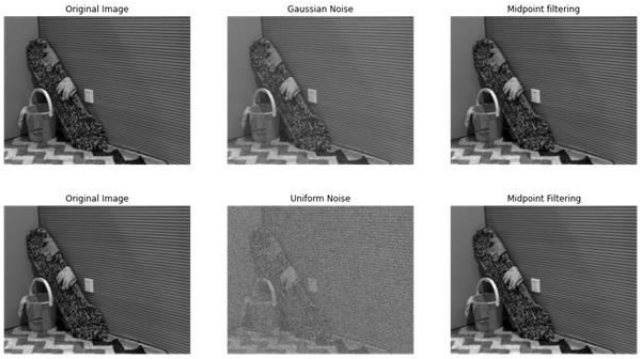
2.8 Gaussian and Uniform noise removal using midpoint filter

A midpoint filter is a type of digital filter that is used to smooth out data by removing noise or outliers. It works by replacing each point in a dataset with the average of that point and its two nearest neighbors. This has the effect of smoothing out fluctuations in the data and making it more predictable.

The midpoint filter is typically used to filter images containing short tailed noise such as Gaussian and uniform type noises. The midpoint filter is defined as :

$$MidPoint(A) = \frac{\min[A(x+i,y+j)] + \max[A(x+i,y+j)]}{2}$$

where the coordinate (x+i, y+j) is defined over the image A and the coordinate (i, j) is defined over the N x N size square mask.

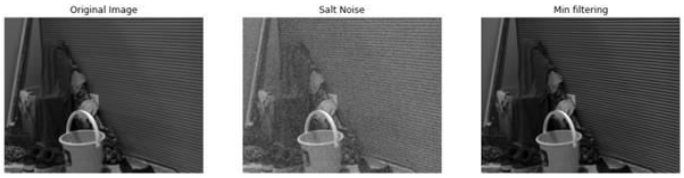


(a)Original Image (b)Gaussian Noise (c)Midpoint Filtering

2.9 Salt Noise removal using Min Filter

When the minimum filter is applied to a digital image it picks up the minimum value of the neighborhood pixel window and assigns it to the current pixel. A pixel with the minimum value is the darkest among the pixels present in the pixel window. The dark values present in an image are enhanced by the minimum filter.

Minimum filter is also called a dilation filter. When a minimum filter is applied the object boundaries present in an image are extended. The minimum filter is typically applied to an image to remove positive outlier noise(salt noise).



(a)Original Image (b)Salt Noise (c)Min Filter

2.10 Pepper Noise removal using Max Filter

The maximum filter replaces each pixel value of a Digital Image with the maximum value(i.e., the value of the brightest pixel) of its neighborhood pixel window. It is the opposite of what the minimum filter does to an Image.

Applying the maximum filter removes the negative outlier noise(pepper noise) present in a Digital Image.

(a)Original Image (b)Pepper Noise (c)Max Filtering

2.11 Inducing Salt and Pepper noise and removing it with median filter

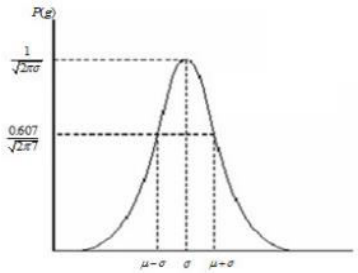
A median filter is a type of digital filter that is used to smooth out data by removing noise or outliers. It works by replacing each point in a dataset with the median of that point and its neighbors. In the context of image processing, this has the effect of smoothing out noisy pixels and reducing the appearance of "salt and pepper" noise in the image.

To implement a median filter in image processing, you would first define the size of the filter window. This is the number of neighboring pixels that will be included in the median calculation for each pixel. A larger window size will result in more smoothing, but may also blur important details in the image.

(a)Original Image (b)Salt and Pepper Noise (c)Median FIltering

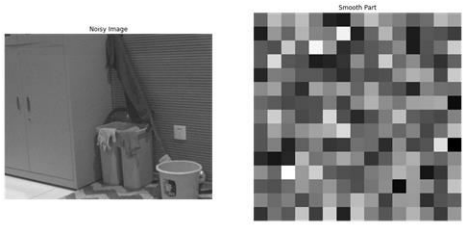
Estimating Probability Density Function of Noise through Histograms

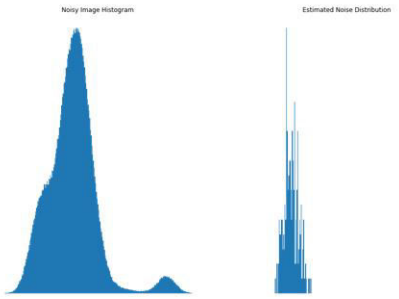
Gaussian Noise is a statistical noise with a Gaussian (normal) distribution. It means that the noise values are distributed in a normal Gaussian way.



The Gaussian noise is added to the original image. The probability density function p of a Gaussian random variable z is calculated by the following formula:

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$





(a)Input Image (b)Smooth Part (c)Noisy Image Histogram (d)Estimated Noise Distribution

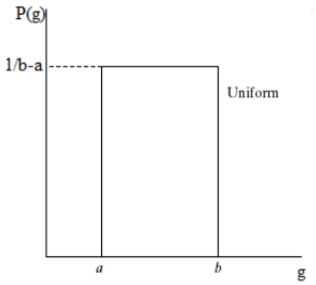
2.12 Uniform Noise

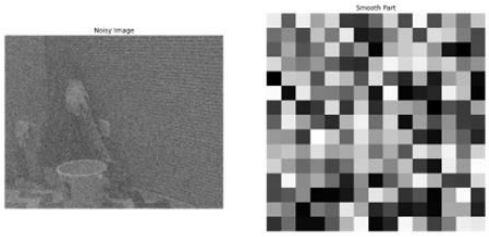
Uniform noise is not often encountered in real-world imaging systems, but provides a useful comparison with Gaussian noise. The linear average is a comparatively poor estimator for the mean of a uniform distribution. This implies that nonlinear filters should be better at removing uniform noise than Gaussian noise.

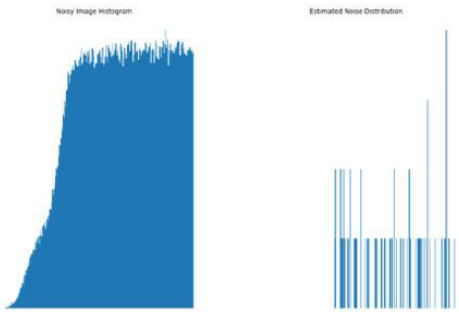
The Uniform pdf is given by:

$$P(g) = \begin{cases} \dfrac{1}{b-a} & \text{if } a \le g \le b \\ 0 & \text{otherwise} \end{cases}$$

The uniform distribution is given below:







(a)Input Image (b)Smooth Part (c)Noisy Image Histogram (d)Estimated Noise Distribution

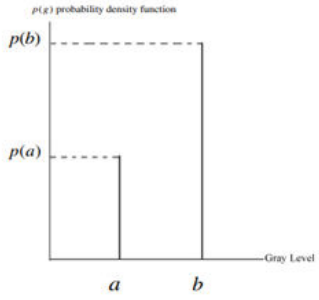
2.13 Salt and Pepper Noise

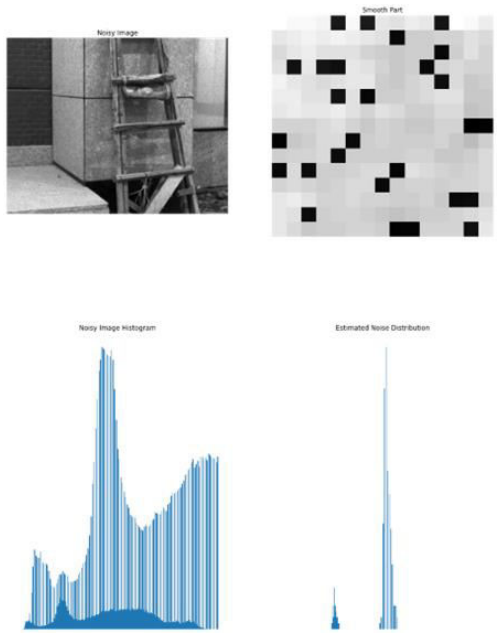
Salt-and-pepper noise, also known as impulse noise, is a form of noise sometimes seen on digital images. This noise can be caused by sharp and sudden disturbances in the image signal. It presents itself as sparsely occurring white and black pixels.

The Uniform pdf is given by

$$P(g) = \begin{cases} Pa \text{ for } g = a \\ Pb \text{ for } g = b \\ 0 \text{ otherwise} \end{cases}$$

The salt and pepper noise distribution is given below:





(a)Input Image (b)Smooth Part (c)Noisy Image Histogram (d)Estimated Noise Distribution

2.14 PSNR (Metrics for quality of denoising effect)

The Peak Signal to Noise Ratio (PSNR) is a measure of the quality of a denoised image in comparison to the original image. This metric is used to measure the noise present in an image that has been subjected to noise induction, as compared to a denoised version of the same image.

	Salt & Pepper Noise	Mean Filter
0	7.853880	10.663167
1	8.445607	11.542024

	Salt Noise	Contra-Harmonic Filter
0	3.225948	11.097712
1	3.350140	11.783453

	Salt Noise	Contra-Harmonic Filter
0	3.225948	11.097712
1	3.350140	11.783453

In general, a higher PSNR value indicates a higher quality reconstructed signal and a lower PSNR value indicates a lower quality reconstructed signal. There is no one optimal PSNR value that applies to all images for this data.

	Periodic Noise	Notch Filtering
0	27.619277	45.344957

	Periodic Noise	Band Reject Filter
0	27.619277	35.088055

The Contra-harmonic mean filter is more effective at denoising the Salt Noise induced image, as indicated by its higher PSNR value compared to the Mean filter removing Salt-Pepper noise induced image.

	Pepper Noise	Max Filtering
0	12.767985	29.684283

	Salt Noise	Min Filtering
0	3.191711	29.470609

	Salt and Pepper Noise	Adaptive Median Filtering
0	11.69656	15.988075

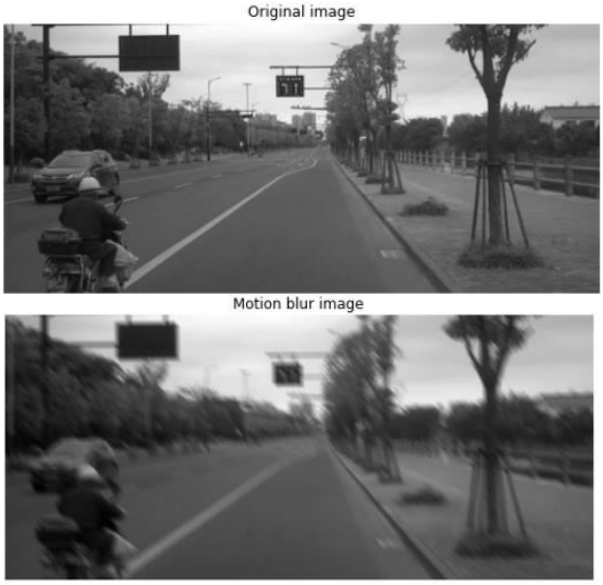
3. Motion Blur and De-Blurring

Motion blur is a common issue in image processing, and it occurs when the camera or subject is moving during the exposure time of the image. This results in an image where objects appear blurred or smeared, as the image captures the movement of the object rather than a static image.

The noise is cleaned using the Wiener filtering method. A gaussian kernel is created (default 3x3 filter) using normal random numbers. That kernel is passed into the Wiener filter method. The constant K is the ratio of Spectral Power Density of the noisy image and the Spectral Power Density of the filter kernel used. We pass it as 10 instead of calculating it manually in the code each time.

Finally after the Fast Fourier Transform function is used, the output must be inverted using Inverse Fast Fourier Transform Function.

$$W(u,v) = \frac{H^*(u,v)}{|H(u,v)|^2 + S_\epsilon(u,v)/S_f(u,v)}$$



(a) Original Image (b) Image after introducing motion blur

(c)Image after de-blurring

4. Periodic noise and removing using Band reject filter

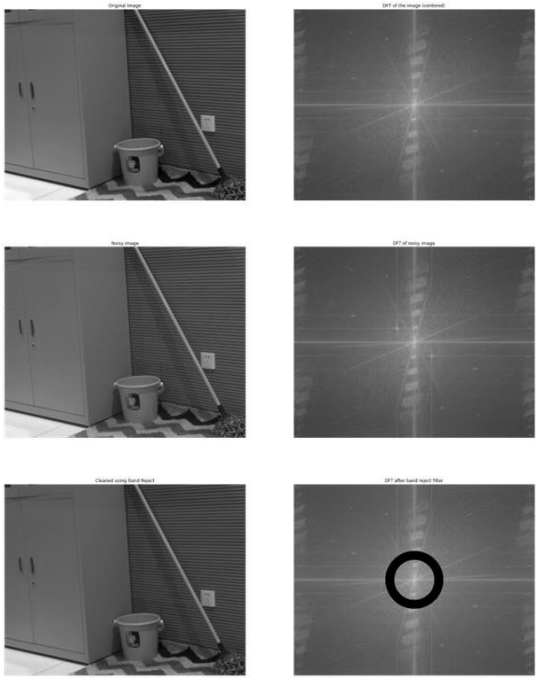
4.1 Periodic Noise

Periodic noise is an unwanted signal that interferes with the source image or signal at a random frequency, depending on its source. Generally, this interference can be added to the image from nature, the electricity network, or electronics devices.

4.2 Band Reject Filter

A band reject filter is useful when the general location of the noise in the frequency domain is known. A band reject filter blocks frequencies within the chosen range and lets frequencies outside of the range pass through.

Band reject filters are ideally suited for filtering out periodic interference. The Fourier transform of a pure sine or cosine function is just a pair of impulses. Therefore the interference is “localized” in the spectral domain and one can easily identify this region and filter it out.

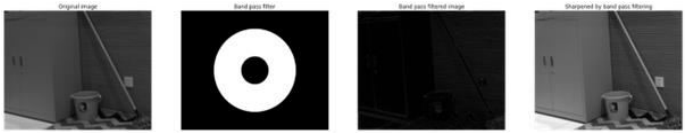


(a)Original Image (b)DFT of image(centered) (c)Noisy image (d)DFT of noisy image (e)Cleaned using Band Reject (f)DFT after band reject filter

4.3 Band Pass Filter

A band-pass filter is a type of image processing filter that is used to remove any image content that falls outside a specified frequency range. It is called a "band-pass" filter because it allows a specific range of frequencies to pass through, while blocking frequencies outside of this range. This can be useful for isolating specific features or patterns in an image, or for removing unwanted noise or clutter.

To use a band-pass filter, you first need to specify the range of frequencies that you want to allow through the filter. This can be done using a lower and upper cutoff frequency. The filter will then remove any image content that falls outside of this range.



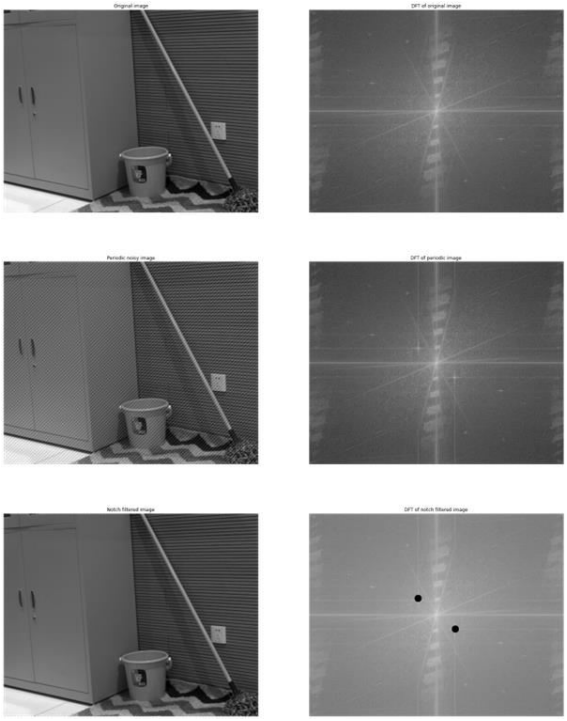
(a)Original Image (b)Band Pass Filter (c)Band Pass Filtered Image (d)Sharpened by Band Pass Filtering

4. 4 Notch Filtering

Notch filtering is a type of frequency-domain filtering that is used to remove or reduce certain frequency

components in a signal or image. It is called "notch" filtering because it creates a "notch" or a dip in the frequency spectrum at a specific frequency or set of frequencies. This can be useful for removing specific types of noise or interference from an image, or for emphasizing certain features in the image.

Notch filters are often used in image processing applications to remove periodic noise, such as noise introduced by a camera's rolling shutter effect or by interference from electrical power sources. They can also be used to remove other types of noise, such as impulse noise or Gaussian noise.



(a)Input Image (b)DFT of original Image (c)Periodic Noise Image (d)DFT of periodic image (e)Notch filtered image (f) DFT of notch filtered image

5. Edge Detection

Edges are the set of pixel positions present in locations of abrupt intensity changes. They represent boundaries between objects with other objects and backgrounds. For detecting edges, there are several filters we can apply to our image.

5.1 Laplacian Filter

This filter uses the property of second derivatives, that at points of onset of ramps / steps, the second derivative value is non zero, while it is zero in regions of constant intensity and on ramps / steps.

In an image, the second derivative for a pixel at position (x, y) can be denoted as:

$$V^2 f = \frac{d^2 f' x}{dx^2} + \frac{d^2 f' y}{dy^2}$$

We can represent this filter by a filter, which can be convoluted on an image to get the Laplacian second derivative mask.

In filter form, the second derivative operator gives us:

[[0, -1, 0], [-1, 4, -1], [0, -1, 0]]

By rotating the kernel by 45 degrees in the counterclockwise or clockwise direction, we obtain a second derivative filter that considers all its 8 Neighbours (when the filter size is (3,3)).

In filter form, the Laplacian n8 is given by:

[[-1, -1, -1], [-1, 8, -1], [-1, -1, -1]

(a) Input Image (b)Image after Laplacian n4 filter (c)Image after Laplacian n8 filter

Analysis

It can be observed that the n8 (I.e., 45 degrees rotated Laplacian kernel) is able to detect edges to a higher degree. This is because for any point (x, y), the Laplacian n8 filter considers all of its pixels in a 8-connectivity region (I.e. (x – 1, y), (x + 1, y), (x, y – 1), (x, y + 1), (x + 1, y + 1), (x – 1, y – 1), (x – 1, y + 1), (x + 1, y – 1)), while the Laplacian n4 filters also considers pixels in a 4-connectivity region (I.e. (x – 1, y), (x + 1, y), (x, y + 1), (x + 1, y)).

Also, the sum of the coefficient of the filter is zero, which indicates that in regions of zero intensity change (I.e. constant intensities), the filter response is net zero.

5.2 Unsharp Masking Sharpening Filter

This sharpening filter works in two steps. First, we blur the image, and then subtract the original image from the blurred image. The resultant is an image of the same dimensions as that of the original image, but with edge details highlighted.

We can enhance the image using the following computations:

$$G(x,y) = f(x,y) + k*(f(x,y) - f'(x,y))$$

Where, G(x, y) is the value at position (x, y) of the enhanced image, F (x, y) is the intensity at position (x, y) of the original source image, K is an integer determining the 'strength' of the unsharp masking filter

F'(x, y) is the intensity as position (x, y) of the blurred source image (any smoothing filter could be used here, and in our project, we have decided to go with a averaging / box blur due to its low computation cost when compared to other spatial domain blurring filters).

The filter works on the basis that regions inside objects remain the same even after a blurring / smoothing operation is applied, while in edge regions (I.e. regions of sudden and abrupt intensity change), the blurred image for any given position (x, y) will give vastly different results compared to the source image at the same position (x, y).

(a)Input Image (b)Unsharp Masking

Analysis

It can be inferred from the above image that the unsharp mask filter gives extremely mild results when compared to the Laplacian sharpening filters.

5.3 Sobel Edge Detector

The Sorbel edge detector approximates the Roberts cross gradient operator (which works based on pixel differentiation/ the gradient).

The gradient at any pixel (x, y) of a source image F is given by

$$Vf(x,y) = [df'/dx, df'/dy]$$

The above-mentioned gradient vector (Vf) points in the direction of greatest rate of change of intensity at some location (x, y). The magnitude of said vector gives us the value of rate of change in the direction the gradient vector points to.

We compute edges for the horizontal and vertical edges separately in two different passes, and combine the Sobel X filter computed image with the Sorbel Y filter computed image to retrieve all edge details of the source image.

The Sobel X Filter (I.e. edge detector in horizontal direction) is given by:

Gx = [[1, 0, -1], [2, 0, -2], [1, 0, -1]

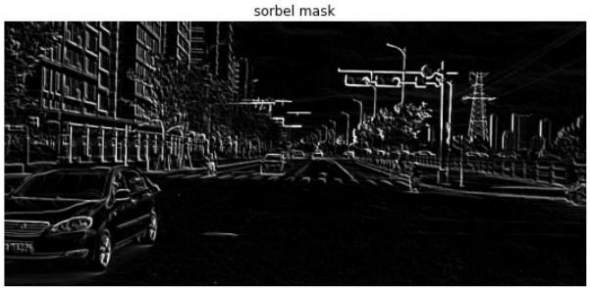
Similarly, the Sorbel Y filter (I.e., edge detector in vertical direction) is given by:

Gy = [-1, -2, -1], [0, 0, 0], [1, 2, 1]

Note how the central pixel (in each column of the Sobel X filter and each row of the Sorbel Y filter) has a higher absolute magnitude when compared to the corresponding adjacent pixels. This is done to give a higher priority to the central pixel as compared to pixels farther than the central pixel (when performing convolution).

Also, the sum of the filter in both the X and Y filters are zero, which indicates that in regions of zero intensity change (I.e. constant intensities), the filter response is net zero.

(a)Input Image (b)Sharpened Image using Sorbel



(c)Sorbel Mask

Analysis:

It can be inferred that the Sorbel edge detector is able to capture almost all the edges present in the source image, while the Laplacian filter gives milder results as compared to Sorbel. For edge detection, either of the two filters (I.e. Laplacian n8 or Sobel edge detector) could be used.

The unsharp masking filter performed rather poorly, and did not detect edges within objects. This is primarily because it does not take into account the magnitude and direction of the gradient or pixel derivative for the reason of faster computation times.

Hence, it can be concluded that the Sorbel X and Y filter are best fit for edge detection in our application, despite the higher computation time (especially since two filters are to be convoluted over our source image, then an additional operation must occur between the two convoluted images).

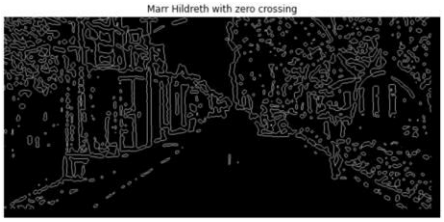
If performance is of concern, we can opt to go with the Laplacian n8 filter, which is less computationally heavy, detects edges with high accuracy, but contains double edges in certain regions.

5.4 Marr Hildreth Edge Detection

The Marr-Hildreth edge detection method is a way to identify the edges of objects in an image. It is based on the idea that the intensity gradient of an image is highest at the edges of objects.

The basic process of the Marr-Hildreth edge detection method involves:

1. Applying a convolution filter to the image to create a smoothed version of the image. This is done to reduce noise and make the edges more distinct.
2. Applying the Laplacian operator to the smoothed image. The Laplacian operator is a second-order derivative operator that is used to detect the intensity gradient of an image. It is sensitive to changes in intensity, so it is able to pick up on the edges of objects.
3. Thresholding the output of the Laplacian operator to create a binary image, where the edges of objects are represented by white pixels and everything else is represented by black pixels.



Analysis:

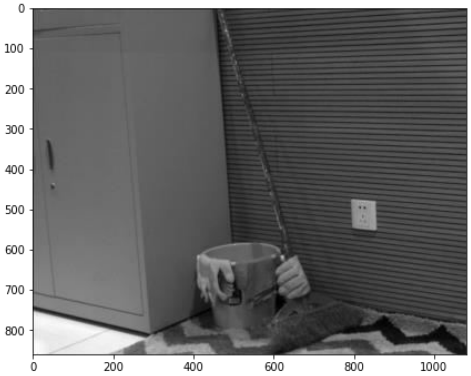
The Marr Hildreth algorithm is able to detect all of the edges properly and as we can see from the image the road appears as a Empty black region and any white pixel in this region can be seen as a disturbance on the road or even a car on the road.

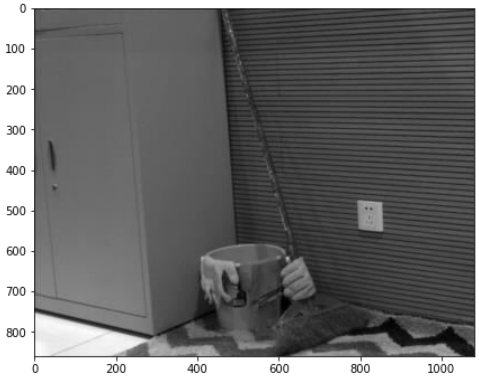
5.5 Canny Edge Detector

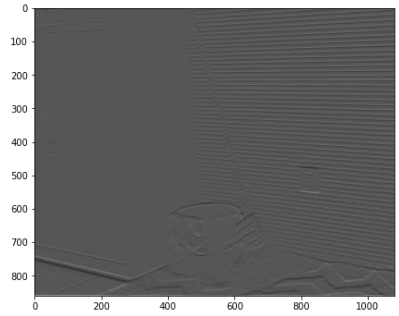
The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of

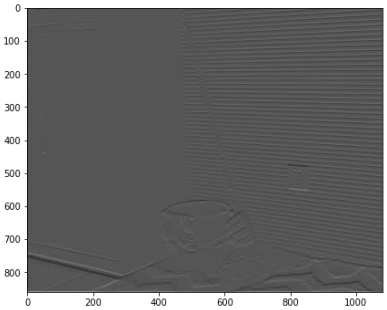
edges in images. The Canny edge detection algorithm is composed of 5 steps:

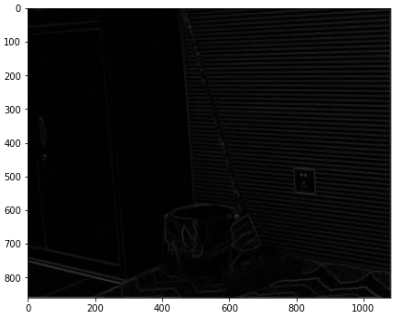
- Noise reduction
- Gradient calculation
- Non-maximum suppression
- Double threshold
- Edge Tracking by Hysteresis

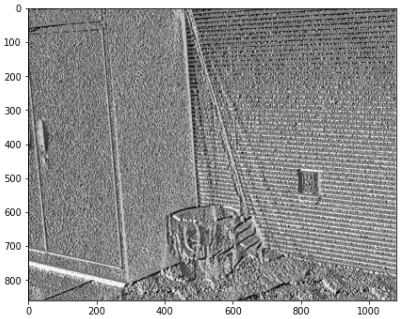


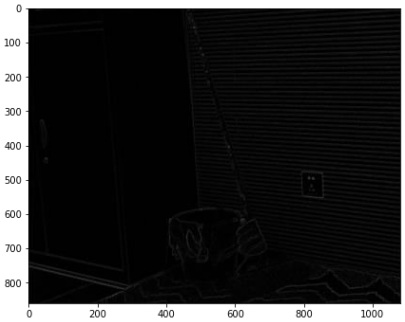


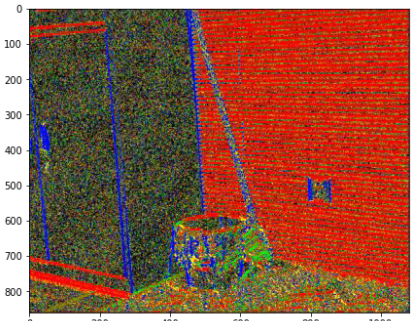


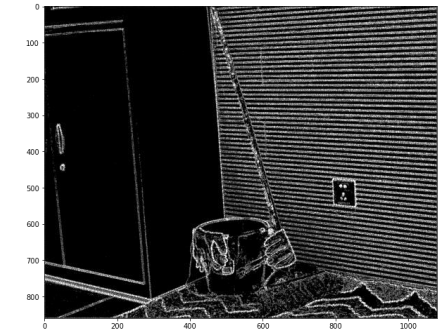


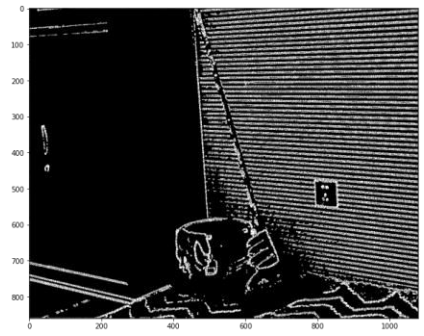












(a) Original image (b) Smoothened image (c) Images after applying gradient mask gx and gy (d) Gradient magnitude (e) Gradient direction (f) Non max suppression (g) Colorized image for visualizing angles (h) Double thresholding, (i) Hysteresis

Analysis

It can be observed from the output that the edge enhancement has been significantly improved over the application of double thresholding and hysteresis concepts. Through double thresholding, strong, weak and non-relevant pixels are identified. Hysteresis mechanism will help in identify the weak pixels that could be considered as strong and the ones that are considered as non-relevant.

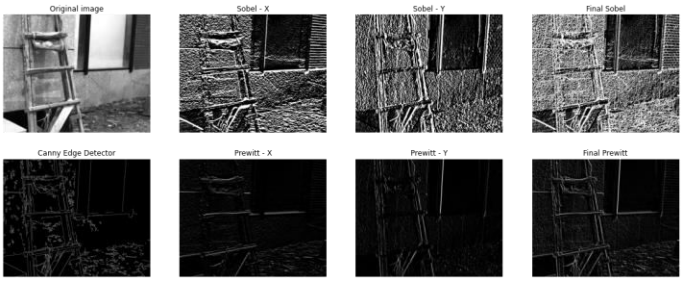
5.6 Sobel and Prewitt Edge Detection

The Sobel is one of the most commonly used edge detectors. It is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. The Sobel edge enhancement filter has the advantage of providing differentiating and smoothing concurrently.

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Prewitt operator is similar to the Sobel operator and is used for detecting vertical and horizontal edges in images. However, unlike the Sobel, this operator does not place any emphasis on the pixels that are closer to the center of the mask.

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$



Analysis

The results of Sobel edge detector shows how abruptly or smoothly the intensity of the point changes and how likely it is the part of the image that represents an edge. The result of Sobel operator at any point of the image is a region of constant image intensity.

The result of Prewitt operator is either the corresponding gradient vector or normal of this vector. It is based on the convolving the image with small, separable and integer valued filter in horizontal(x) and vertical(y) direction. It is computationally less expensive and faster method for edge detection.

5.7 . Morphological Gradient Operator

A simple morphological image processing technique used to find the borders / outline of images.

Consist of two steps:

(i) Perform erosion to shrink the image based on the structuring element.

(ii) Subtract eroded image from the dilated image.

(a) Source Image (b) Result of morphological gradient operator

Analysis

The Gradient morphological operator performs well for certain images and can be used as an alternative for edge detection, in cases where double edges are not a problem

and perfect edge detection (such as by using the sobel filter) is acceptable. It can be considered to be a method of segmentation.

6. Filtering Processes (Low / High Pass Filters)

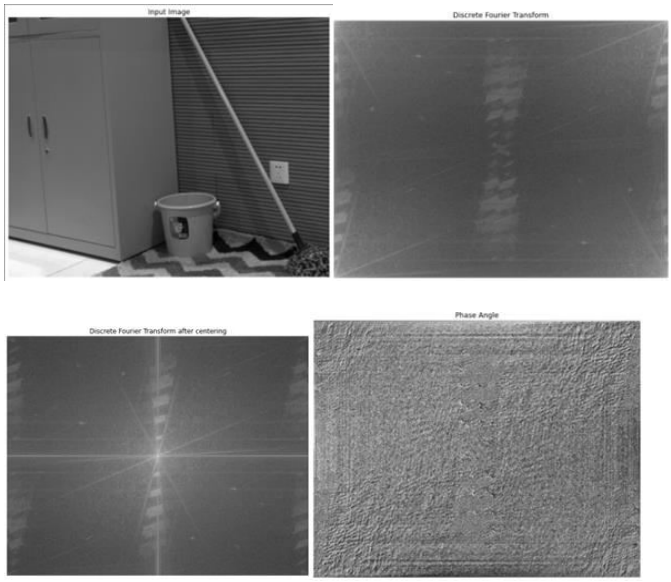
6.1 Gaussian Low Pass and High Pass Filter

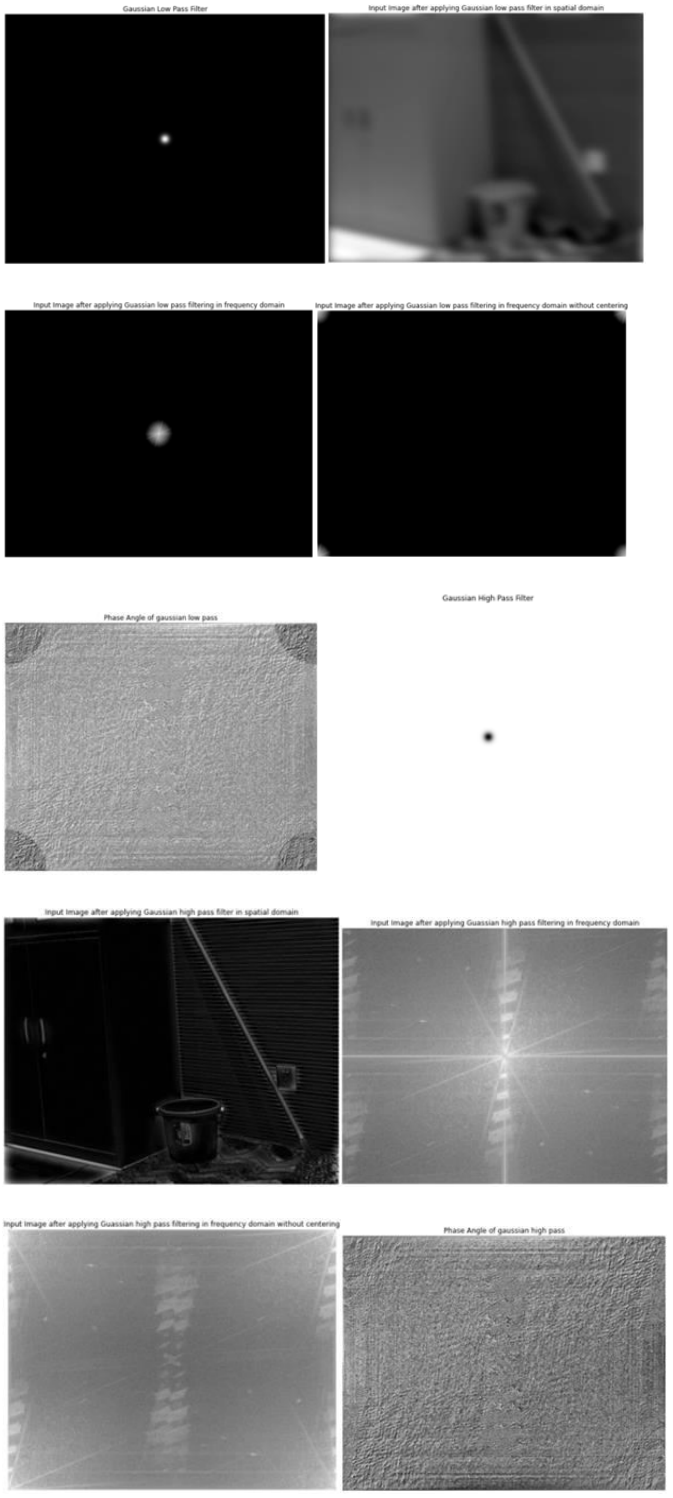
A Gaussian low pass filter is a filter that allows low frequency signals to pass through it while attenuating high frequency signals. This type of filter is often used to smooth images or remove noise from signals. It works by convolving the input signal with a Gaussian kernel, which is a bell-shaped curve that decays rapidly as the distance from the center increases.

A Gaussian high pass filter is the opposite of a low pass filter, in that it allows high frequency signals to pass through while attenuating low frequency signals. It is often used to sharpen images or to highlight features in a signal. Like the low pass filter, it works by convolving the input signal with a Gaussian kernel, but in this case the kernel has a negative value in the center and positive values at the edges.

Both types of filters are named after the Gaussian function, which is a continuous, smooth curve that is used to model the probability distribution of many real-world phenomena. The Gaussian function has the property of being symmetrical about its center, which makes it well suited for use in filters that need to be symmetrical.

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2}$$





(a)Input Image (b)Discrete Fourier Transform (c)Discrete Fourier Transform after centering (d)Phase Angle (e)Gaussian Low pass Filter (f)Images after applying Gaussian Low Pass Filter in Spatial Domain (g)Images after applying Gaussian Low Pass filter in Frequency Domain (h)Images after applying Gaussian Low Pass filter in Frequency Domain without centering (i)Phase Angle of Gaussian Low Pass (j)Gaussian High Pass Filter (k)Input Image after applying Gaussian High Pass filter in Spatial Domain (l)Input Image after

applying Gaussian High Pass filter in Frequency Domain (m)Input Image after applying Gaussian High Pass filter in Frequency Domain without centering (n)Phase Angle of Gaussian High Pass

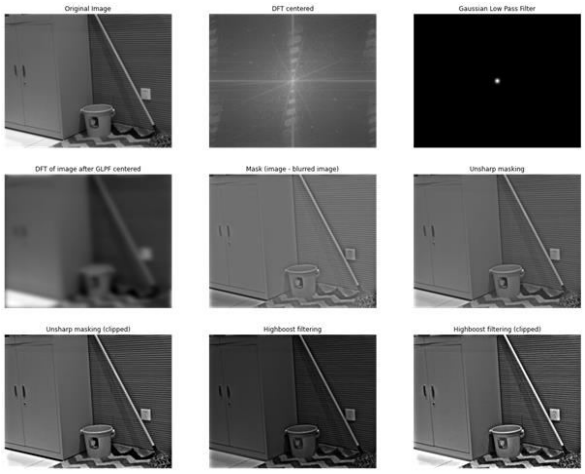
6.2 Unsharp Masking and High Boost Filtering

Unsharp masking

In unsharp masking, the blurred version of the image is taken as a mask which is then subtracted from the original image. This gives a clearer image as the initial blur is removed.

High-boost filtering

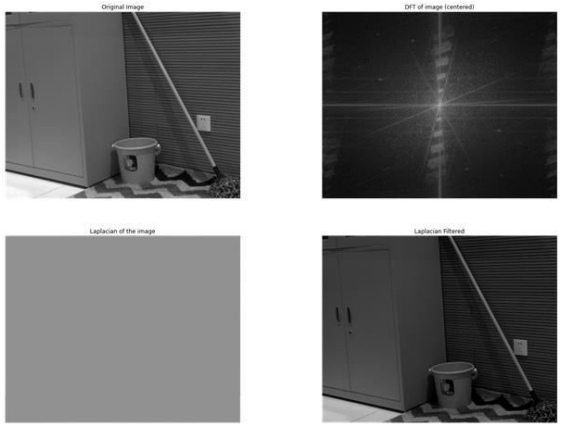
In image processing, it is often desirable to emphasize high frequency components representing the image details without eliminating low frequency components (such as sharpening). The high-boost filter can be used to enhance high frequency components.



(a)Original Image (b)DFT Centered (c)Gaussian Low Pass Filter (d) DFT of image after GLPF centered (d) Mask (image - blurred image) (e)Unsharp masking (f)Unsharp masking(clipped) (g) Highboost Filtering (h) Highboost filtering(clipped)

6.3 Laplacian Filter

Laplacian filter is a second-order derivative filter used in edge detection, in digital image processing. In 1st order derivative filters, we detect the edge along with horizontal and vertical directions separately and then combine both. But using the Laplacian filter we detect the edges in the whole image at once.



(a)Original Image (b)DFT of image(centered) (c)Laplacian of the image (d)Laplacian Filtered

Frequency domain filters

In the frequency domain, once the image has been centered, it must be noticed that the low frequency components are present close to the center, and as we go away from it, the high frequency components dominate. Since the edges are associated with high frequency components, the frequency domain can be used to efficiently enhance / cull certain parts of an image by frequency.

i.e.,by attenuating regions close to the image's center, we can retrieve only the high or low frequency components of the image.

6.4 Ideal High Pass Filter

This filter works by attenuating low frequency components around the image center. The filter is mathematically defined as:

$$H(u,v) = \begin{cases} 1\,, if\ D(u,v) \geq D_0 \\ 0\,, if\ D(u,v) < D_0 \end{cases}$$

Here, D(u, v) is given by:

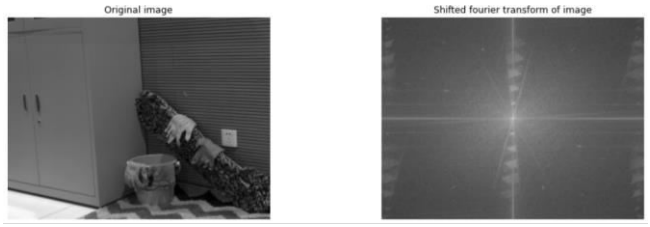
$$\sqrt{(u - M/2)^2 + (v - N/2)^2}$$

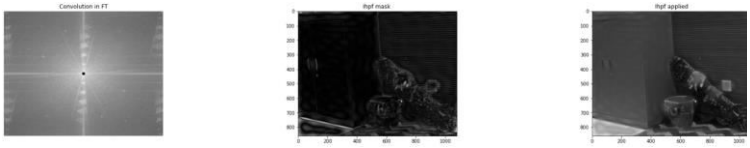
where,

u, v : coordinate of image in frequency domain, M, N : width and height of the source image (in pixels), D_0 is the cutoff frequency of the image (or the radius of the circle encompassing the low frequency components of frequency domain image)

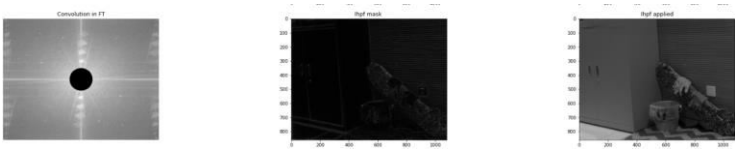
The resultant image in the frequency domain enhances the edges from the source image while losing all intensity / contrast information. The resultant spatial domain image has significant banding effect. This is primarily because our high pass filter was designed in the frequency domain, but while converting from said domain to spatial domain, we apply the inverse Fourier transform.

The transformation from frequency domain to spatial domain involves using the inverse fast Fourier transform (FFT) which involves the integral of weighted average of sines and cosines. The ringing effect is observed due to this phenomenon.





(a) Convolution in Fourier Transform (b)Ideal High pass Filter mask (c)Image after applying Ideal high Pass filter



(b) Convolution in Fourier Transform (b)Ideal High pass Filter mask (c)Image after applying Ideal high Pass filter

An image showing the result of Ideal high pass filter in the spatial domain.

It can be noted that as the cutoff frequency D0 increases, the ringing effect is heavily subdued, but the resultant spatial domain and frequency domain image loose significant edge / high frequency details.

6.5 Ideal low pass filter

The Ideal Low pass filter is simply obtained by performing 1 – IHPF, which means:

The high frequency components in the frequency domain of source image are attenuated, leaving behind only the low frequency components (I.e. components associated with intensity / contrast).

The filter is obtained by the computation:

$$H(u,v) = \begin{cases} 0, if D(u,v) < D_0 \\ 1, if D(u,v) < D_0 \end{cases}$$

Here, D(u, v) is given by:

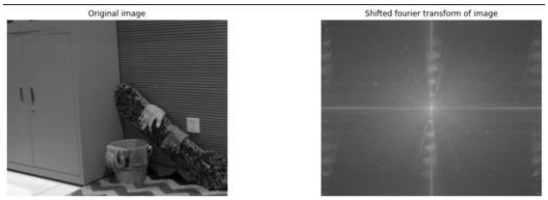
$$\sqrt{(u - M/2)^2 + (v - N/2)^2}$$

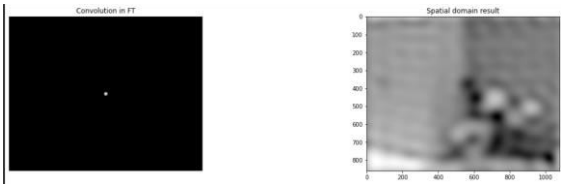
Where,

u, v : coordinate of image in frequency domain, M, N : width and height of the source image (in pixels), D_0 is the cutoff frequency of the image (or the radius of the circle encompassing the low frequency components of frequency domain image)

The resultant image is heavily blurred as compared to the source spatial domain image. This is because we attenuate the high frequency components (which correspond to edges), leaving behind only low frequency components. The resultant spatial domain image has significant banding effect. This is primarily because our high low filter was designed in the frequency domain, but while converting from said domain to spatial domain, we apply the inverse Fourier transform.

The transformation from frequency domain to spatial domain involves using the inverse fast Fourier transform (FFT) which involves the integral of weighted average of sines and cosines. The ringing effect is observed due to this phenomenon.





(a)Original Image (b) Shifted Fourier Transform of image (c)Convolution in Fourier Transform (d)Spatial Domain result

An image showing the result of Ideal low pass filter in the spatial domain.

6.6 Butterworth Low Pass Filter

This is a frequency domain low pass filter which gives us more control and flexibility compared to the ideal low pass filter.

Mathematically, this filter is given by:

$$H(u,v) = 1/(1 + (D/D_0)^{2n})$$

Where, D is given by:

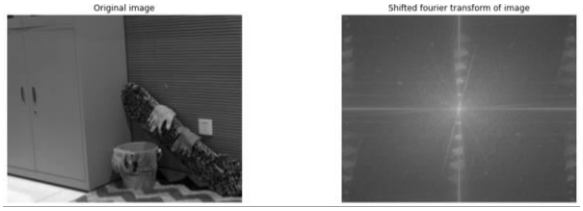
$$\sqrt{(u - M/2)^2 + (v - N/2)^2}$$

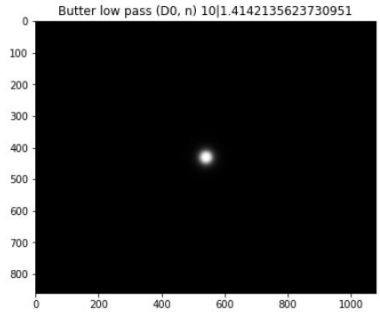
D_0 = cut off frequency, n = order of filter, U, v : coordinate of image in frequency domain, M, N : width and height of the source image (in pixels)

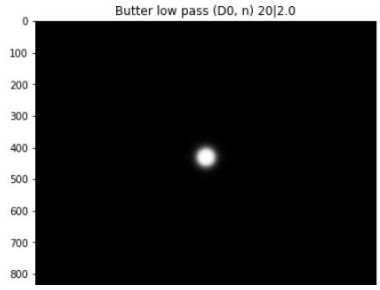
It can be noticed that as the value of n (the order of filter) approaches infinity, the filter acts similar to the ideal low pass filter, which produces high blurring effects with significant ringing, which can be undesirable.

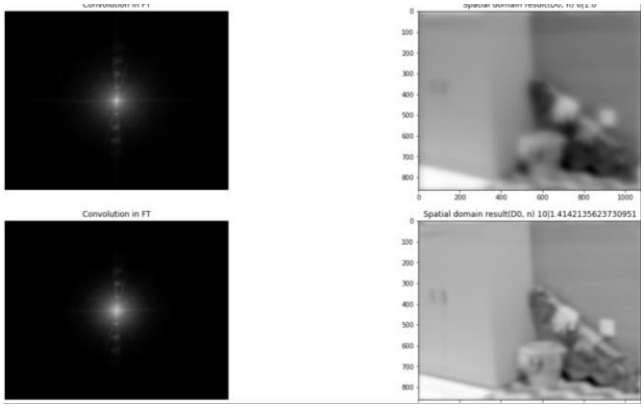
Also, when the order of filter approaches zero, the effect of butter worth filter is similar to that of the gaussian low pass frequency domain filter, where blurring is present with little to no ringing effect.

In most applications, an order of 2 gives the best ringing effect to blurring effect balance.









It can be observed that the butter worth low pass filter for low orders (i.e. order n < 2) produces a halo effect : This is in contrast to the extremely sharp cutoffs as seen in the ideal low pass filters.

6.7 Butterworth High Pass Filter

This is a frequency domain low pass filter which gives us more control and flexibility compared to the ideal high pass filter. It is obtained by performing 1 – Butterworth Low Pass Filter.

Mathematically, this filter is given by:

$$H(u,v) = 1/(1 + (D0/D)^{2n})$$

Where D is given by:

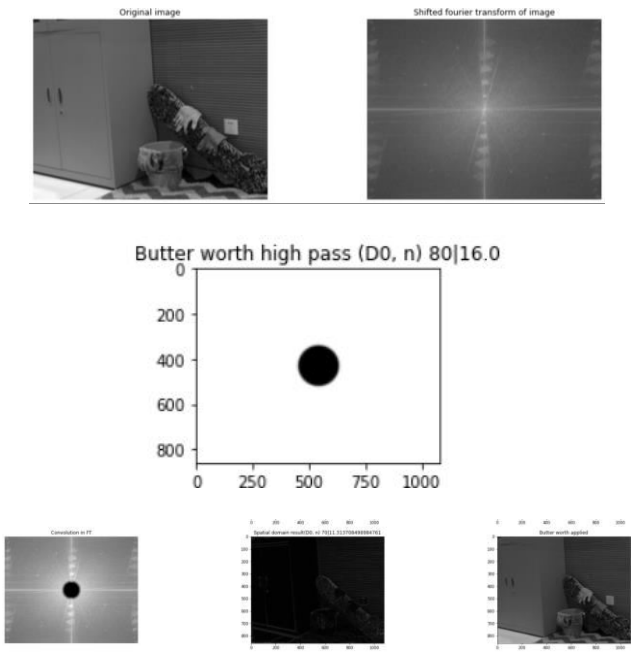
$$\sqrt{(u - M/2)^2 + (v - N/2)^2}$$

D0 = cut off frequency, n = order of filter, U, v : coordinate of image in frequency domain, M, N : width and height of the source image (in pixels)

It can be noticed that as the value of n (the order of filter) approaches infinity, the filter acts similar to the ideal high pass filter, which attenuates low frequency components leaving behind the high frequency components (present far away from the center of the image in frequency domain). blurring effects with significant ringing, which can be undesirable.

Also, when the order of filter approaches zero, the effect of butter worth filter is similar to that of the gaussian high pass frequency domain filter, where blurring is present with little to no ringing effect.

In most applications, an order of 2 gives the best ringing effect to sharpening effect balance.



(a)Convolution in Fourier Transform (b)Spatial Domain Result (c)Butterworth applied

It can be observed that the butter worth low pass filter for low orders (I.e. order n < 2) produces a halo effect : This is in contrast to the extremely sharp cutoffs as seen in the ideal high pass filters.

Analysis:

In the frequency domain, the butter worth filter (for low orders of n, where n < 5) produces the best sharpening / blurring filter with the flexibility of choosing between ringing effect and blurring / sharpening. The ideal low / high pass filters, though easier to compute, produce a very unfavorable ringing effect (when some ringing effect is allowable, we can extract more high frequency / low frequency components).

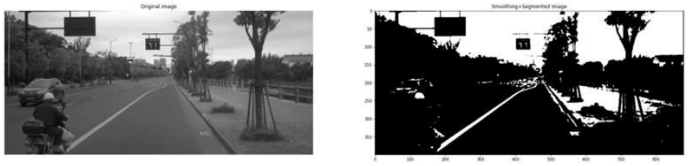
7. Segmentation

Smoothing and Segmentation

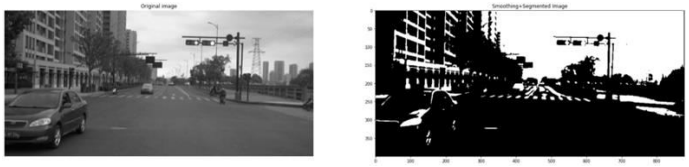
Smoothing is a process that aims to reduce the level of detail in an image, often by removing noise or reducing the contrast between neighboring pixels. This can be useful for eliminating distractions or making an image easier to interpret. Segmentation, on the other hand, is the process of dividing an image into distinct regions or segments, each of which corresponds to a different object or background in the image. This can be useful for identifying and analyzing specific objects or features in an image. Segmentation can be performed using a variety of methods, including thresholding, clustering, and edge detection.

Here thresholding has been used as a form of segmentation to segment pixels with values greater than 127 and pixels with values lesser than 127.

Together, smoothing and segmentation can be used to improve the clarity and interpretability of an image, as well as to extract useful information or features from it.



From Left (a),(b): (a) Input Image (b)Smoothened and Segmented Image



From Left (a),(b): (a) Input Image (b)Smoothened and Segmented Image

7.1 Low and High Thresholding

Thresholding is an image processing technique that is used to convert a grayscale image into a binary image. A binary image is an image in which each pixel is either black or white, with no shades of gray. Thresholding is often used to segment an image, that is, to divide the image into distinct regions or objects based on the intensity values of the pixels.

Low thresholding is a technique in which pixels with intensity values below a certain threshold are set to black, and pixels with intensity values above the threshold are set to white. This results in a binary image in which the objects of interest are represented as white pixels, and the background is represented as black pixels.

High thresholding is similar to low thresholding, but the roles of the black and white pixels are reversed. In high thresholding, pixels with intensity values above a certain threshold are set to white, and pixels with intensity values below the threshold are set to black. This results in a binary image in which the background is represented as white pixels, and the objects of interest are represented as black pixels.

From left to right (a)Low Threshold Image(b)High Threshold Image

From left to right (a)Low Threshold Image (b)High

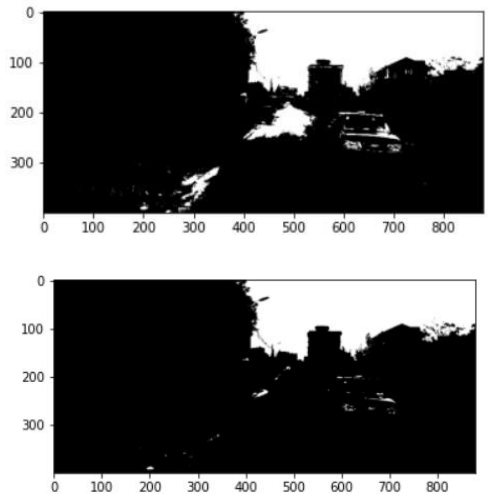
Threshold Image

.7.2 Smooth and Hard Thresholding

Smooth thresholding and hard thresholding are two types of thresholding techniques that are used to convert a grayscale image into a binary image.

In smooth thresholding, the transition from black to white pixels is not abrupt, but rather occurs gradually over a range of intensity values. This results in a smoother, more gradual transition between the foreground and background pixels in the binary image. Smooth thresholding is often used when the intensity values of the pixels in the image are not clearly separated, and a more gradual transition is needed to accurately segment the image.

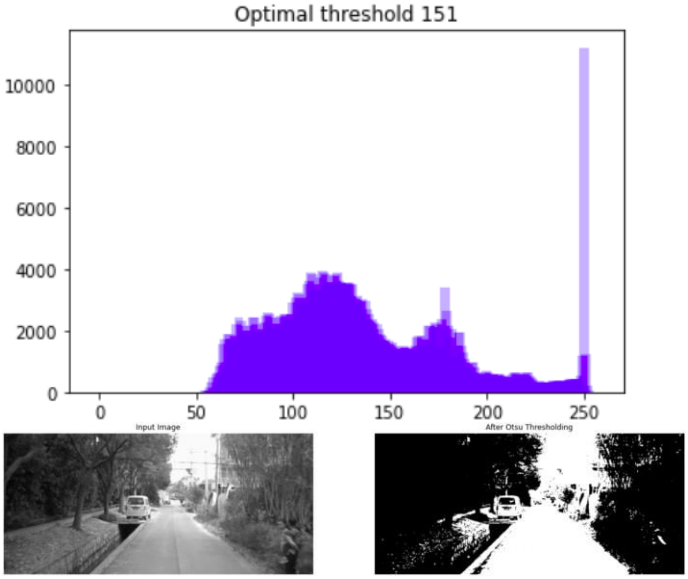
Hard thresholding, on the other hand, involves an abrupt transition from black to white pixels at a certain intensity value. Pixels with intensity values below the threshold are set to black, and pixels with intensity values above the threshold are set to white. This results in a binary image with a clear separation between the foreground and background pixels. Hard thresholding is often used when the intensity values of the pixels in the image are well-separated and an abrupt transition is sufficient to accurately segment the image.



From Left to Right (a)Hard Thresholding (b)Smooth Thresholding

7.3 OTSU Global Threshold

The image is plotted as a histogram and the two separable components in such a histogram will be the foreground pixels and the background pixels. Otsu's algorithm deals with finding a global threshold K from the histogram of a gray scale image. The algorithm uses two statistical measures: within class variance and between class variance to find the optimal global threshold.



(a) Histogram of gray scale image (b) Source image (c) Thresholded image using OTSU.

Analysis

From the histogram it can be inferred that, intensity value of 156 is the middle point demarcating the foreground and the background pixels. Hence this intensity value is chosen as the optimal threshold value. Since this is a global thresholding technique, the inside scene elements are not present on the threshold image. Only the foreground and the background are segmented.

7.4 Improving Otsu's Global Thresholding by image smoothening

Using any sort of image smoothing technique (Gaussian filtering, Average filtering, etc.) then applying Otsu results in a better segmentation of foreground and background pixels as noise will be either removed or suppressed in the smoothened image.

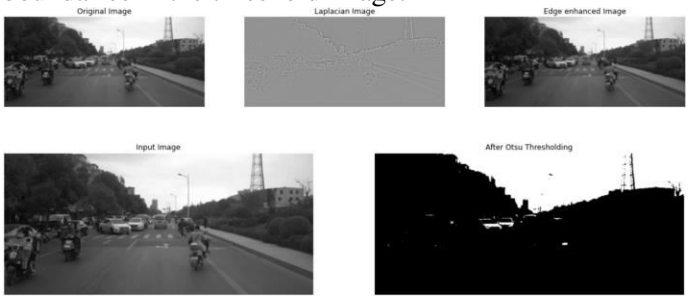
(a, c) Original image (b) Smooth image (C) OTSU threshold

Analysis

Compared with the original image, the smoothened image has lesser noise hence better segmentation of background and foreground pixels

7.5 Improving Otsu's Global Thresholding by Edge enhancement

Using any edge detection techniques (Canny, Laplacian, Maar Hildreth, etc.) the edges in the image can be detected and enhanced. Applying Otsu's Global Thresholding on edge enhanced image results in stronger boundaries in the threshold image.



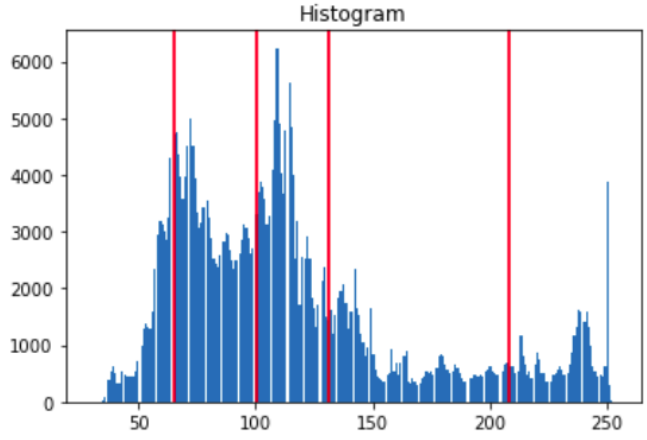
(a) Original image (b) Laplacian image (c) Laplacian enhanced image (d) input image (e) OTSU threshold image

Analysis

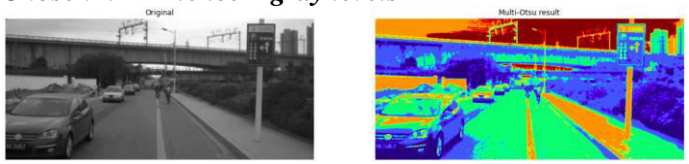
Compared with the original image's threshold the edge enhanced image's threshold contains strong boundary between foreground and background elements.

7.6 Multilevel Thresholding using Otsu's Thresholding

Multilevel thresholding is a process that segments a gray level image into several distinct regions. A value "n" can be fed into the algorithm and "n" number of gray levels will be present in the threshold image.



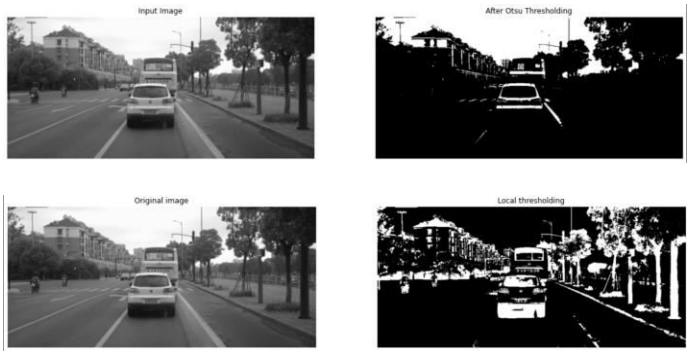
Chosen n = 4 hence 4 gray levels



(a) Gray scale image histogram, (b) Source image © Multi OTSU threshold result

7.7 Variable Thresholding

Applying thresholding techniques on a local neighborhood using a local threshold value is the concept behind variable thresholding. Applying local thresholding on images results in segmentation of both foreground and background images along with in-scene objects based on their relative depth.



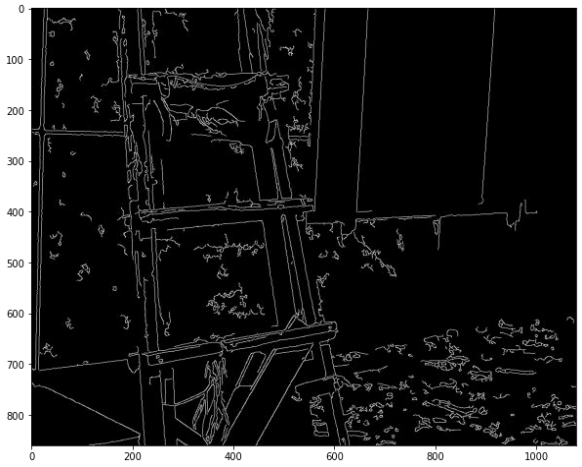
(a) Source image (b) OTSU threshold © Source Image, (d) Local threshold

Analysis

Global thresholding does not segment the objects in a scene but focuses on background and foreground pixels but local thresholding segments all the objects in a scene based on relative depth inside a local region.

7.8 Global Thresholding using Hough Transform

The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc



Analysis

If there are too many lines in the result, or too few of them, we should adjust rho resolution, theta resolution and threshold parameters. But tuning parameters which are used in blurring and edge detection should also be taken into consideration while making the algorithm. Whole pipeline should be revised to fit the needs of the use case.

7.9 Color Space Segmentation (HSV)

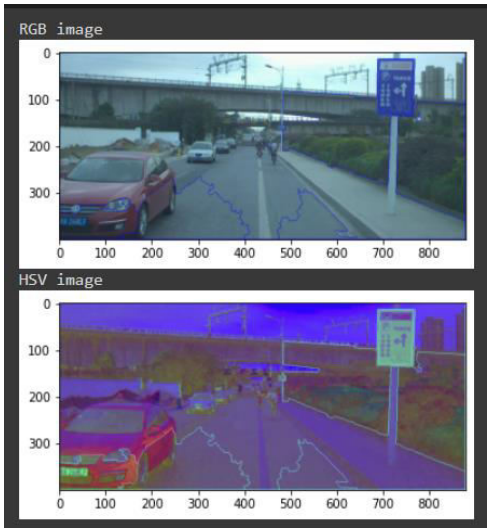
Given an RGB image, we first convert the color space to HSV (Hue Saturation Value). The reason for this is :

(i) Hue: Gives only color information (based on degrees, where red falls between 0 to 60, yellow falls between 61 to 120, etc).

(ii) Value: The darkness / brightness of the color.

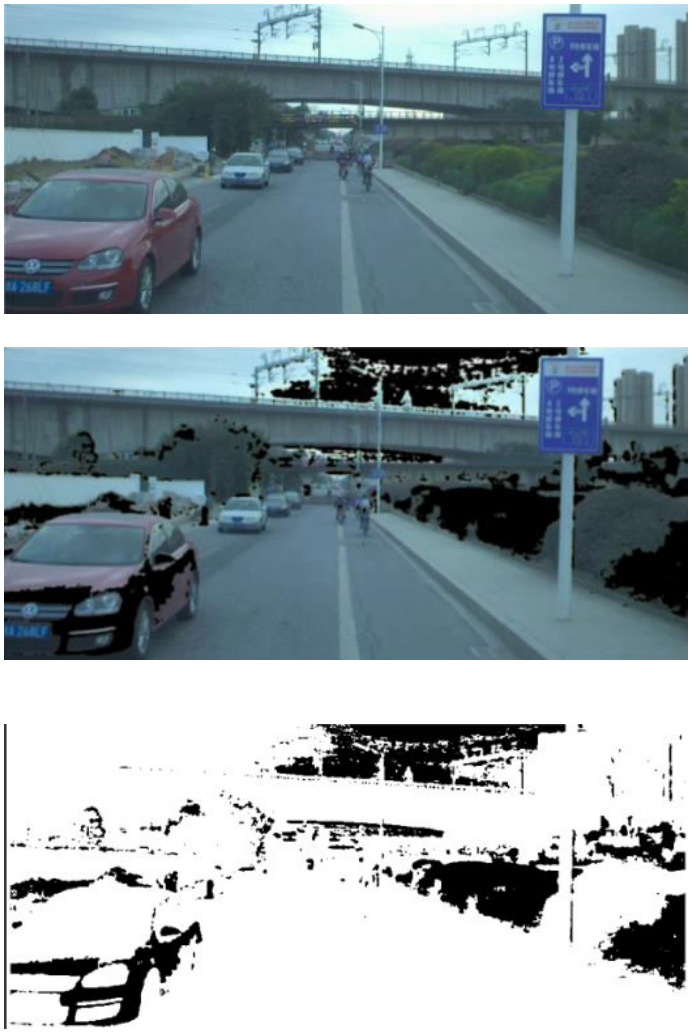
(iii) Saturation: The amount of gray in the image. If this value is 0%, it produces a faded effect and produces whiter. On the other hand, if the value is 100%, we get the primary color itself.

If we are able to retrieve the Hue value alone from a given HSV image, we can segment the objects based on just the color information, disregarding the shade / brightness of the *same* color present in several objects over the scene.

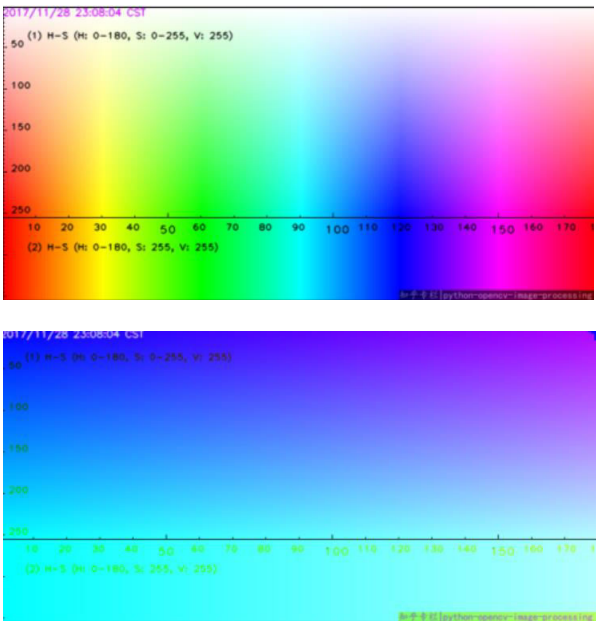


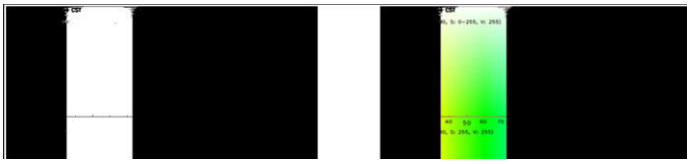
(a) Source image in RGB (b) HSV image

(a) Source color image (b) Segmented image (for cyan - green band in HSV range (72, 0, 0), (90, 255, 255) (C) Segmentation Mask



(a) Source color image (b) Segmented image (for blue band in HSV range (140, 0, 0), (170, 255, 255) © Segmentation Mask





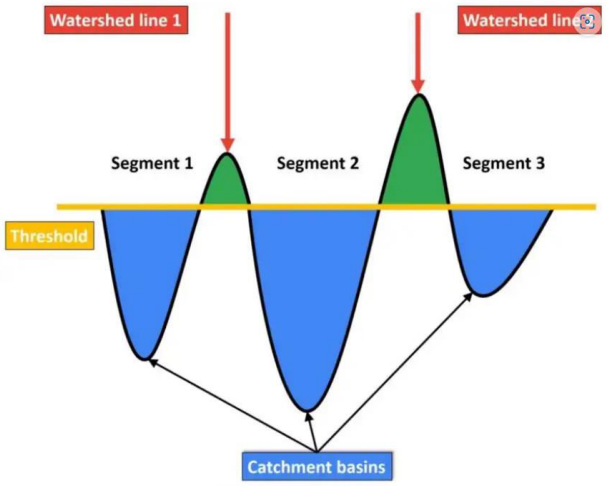
(a) Source RGB Image, (b) HSV converted source image © Segmented image (for green band in HSV range (35, 0, 0), (72, 255, 255)

Analysis

Segmentation by color offers several advantages such as the ability to detect signal lights (Red, Green, or yellow) that the autonomous vehicle system can use to perform the determined action.
This can also be used for detection of stop signs, which are typically colored in specific predetermined formats.

7.10 Watershed Segmentation

In geographical sense: An array where water accumulates. It is a land area that channels / drains rain and snow to creeks, streams, etc. Watersheds can be segmented as topographical maps with boundaries (lower the altitude of the land, more water it accumulates). I.E, the gray scale images can be considered to have valleys and peaks, based on the intensity. There is also a threshold that segregates between the two. The brightness determines the high (peak) and low (valley) in this algorithm.
It works by filling values (i.e., local minima), with pixels belonging to the same predefined label. High intensity denotes peaks and hills, while low intensity denotes valley. We fill every local isolated minimum and mark it as a different segment. We also create a boundary / barrier between two isolated valleys.
The algorithm is useful for segmenting images into background and foreground for certain images where other algorithms fail, for example when the foreground has several objects of same / similar intensity, other algorithms will determine that all said objects will be merged to one, while the watershed algorithm prevents this.
However, it fails when RGB image is used (primarily because if objects are not clearly differentiable when we do the thresholding of image). Also, noise can heavily corrupt / bias the result, for which denoising is required as a preprocessing step.



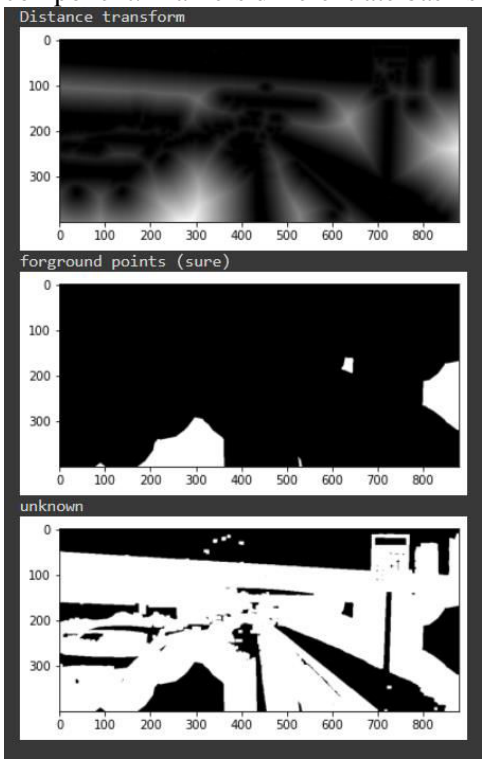
(a) Visualization of gray scale image as topological map

Threshold based segmentation can be used to determine which parts of an image are surely part of the background. OTSU thresholding gives best results in this case.
To determine which objects are surely in the foreground, we can perform erosion to make pixels around boundaries thinner. However, smaller objects tend to get eroded away. To prevent this, we opt for the Distance Transform,

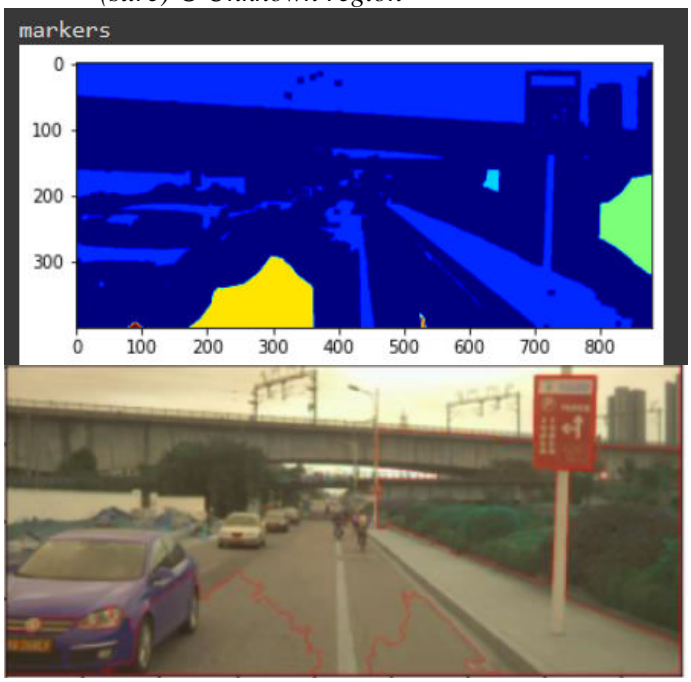
(a) Source Image (b) Definite background (0's in thresholded image)

The distance transform indicates the minimum distance to the closest background (to the region of zeros in the thresholded image).
We will also have Unknown regions just the difference between surely foreground and surely background. We cannot necessarily determine whether objects in the Unknown region belong to the foreground or the background.

We can then find the connected components (usually using 8 connectivity) and get markers for each connected component. Markers differentiate basins from each other.



(a) Distance transform result (b) Foreground points (sure) © Unknown region



(a) Markers (different basins represented by various colors)(b) Watershed result..

Analysis:

We can observe that the watershed algorithm does not work extremely well. This is because the source image itself is a color image, and performing threshold-based segmentation on color images does not produce accurate results.

In images where the background has rather constant intensity and the foreground objects have intensity vastly different from the background, the watershed algorithm works extremely well.

8. Morphological Transformations

Broad set of image processing operations that processes binary images based on SE (Structuring elements) which dictates the nature of the operation.

Each pixel in the image is adjusted based on the value of pixels in its neighborhood.

It is not subjective. Used for getting the ground truth of the description of region shapes, boundaries, and for extracting the various components from the image.

8.1 Erosion

It is a morphological image processing technique that is used to reduce the size of bright regions in an image. This process erodes the boundaries of these regions in the image. It is typically used to remove noise or small objects from an image.

Pixel will be considered 255 or 0 only if ALL pixels under the kernel are 255 or 0. Otherwise it is eroded (i.e color changes to 0 or 255)

Mathematically,

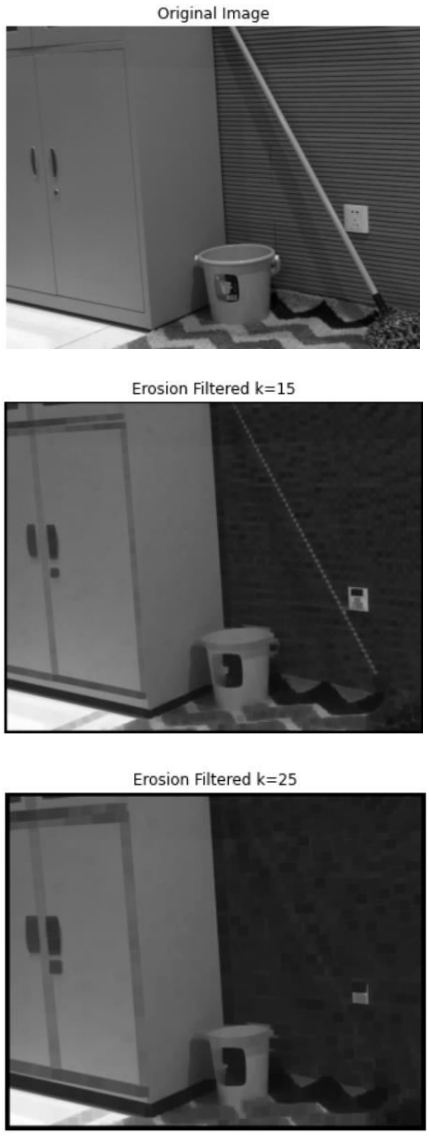
If A is the set and B is the kernel / structural element, then, the erosion of A by B is denoted by A (-) B, where

$$A \ominus B = \{z \mid (B)z \text{ is contained in } A\}.$$

For the Structural element to erode the image at any point, the ENTIRE structuring element must be contained by the binary image at said point.

It shrinks or thins objects in a given binary image, as it removes pixels on object boundaries. Enlarges foreground holes, remove irrelevant small details from the image.

It is also extremely useful for removal of isolated pixel and thin edge boundaries (useful for separation of two objects that are sticking together).



(a)Original Image (b)Erosion after 1 iteration (c)Erosion after 15 iterations (d)Erosion after 25 iterations

8.2 Dilation in image processing

Used for 'expanding an element A by some structural element B'. Adds pixels to object boundaries.

Dilation is a morphological image processing technique that is used to increase the size of the bright regions in an image. This process is used to add pixels to the boundaries of objects in an image. It is used to connect or join the disconnected objects.

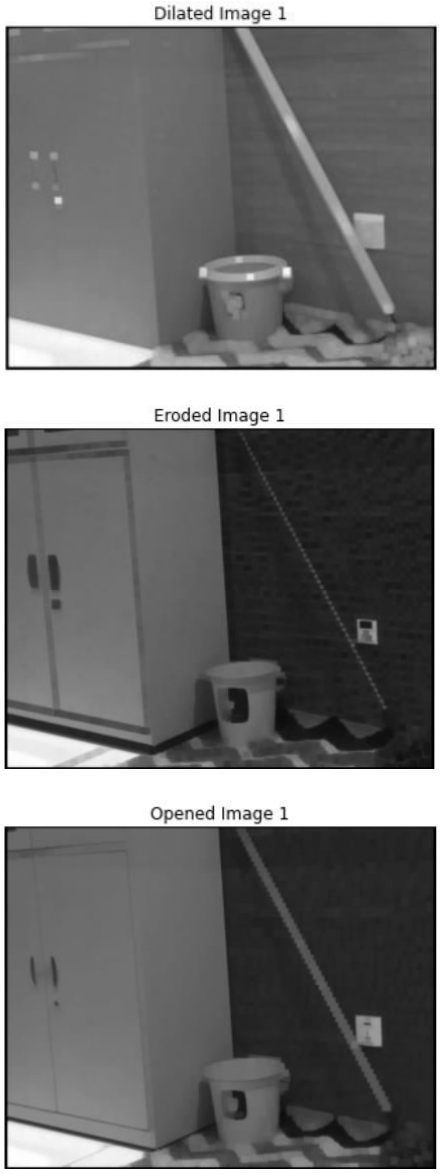
Even if a single pixel of the image is covered by the structural element, we set the entire area covered by the structural element to 1, else 0.

(a)Original Image (b)Dilation done with 5 iterations (c)Dilation done with 6 iterations

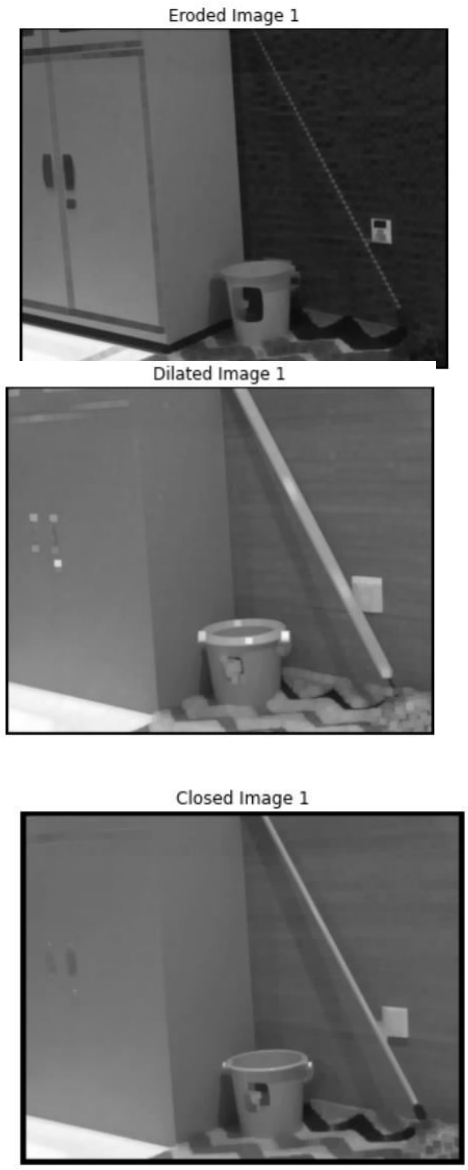
8.3 Opening

The Opening is a process of removing small objects from the foreground of an image and placing them in the background. This process, which consists of dilation followed by erosion, modifies the image.

$$(A \bullet B) = (A \oplus B) \ominus B$$



(a)Dilated Image (b) Eroded Image (c)Opening of an image.



(a)Eroded Image (b) Dilated Image (c)Closed image.

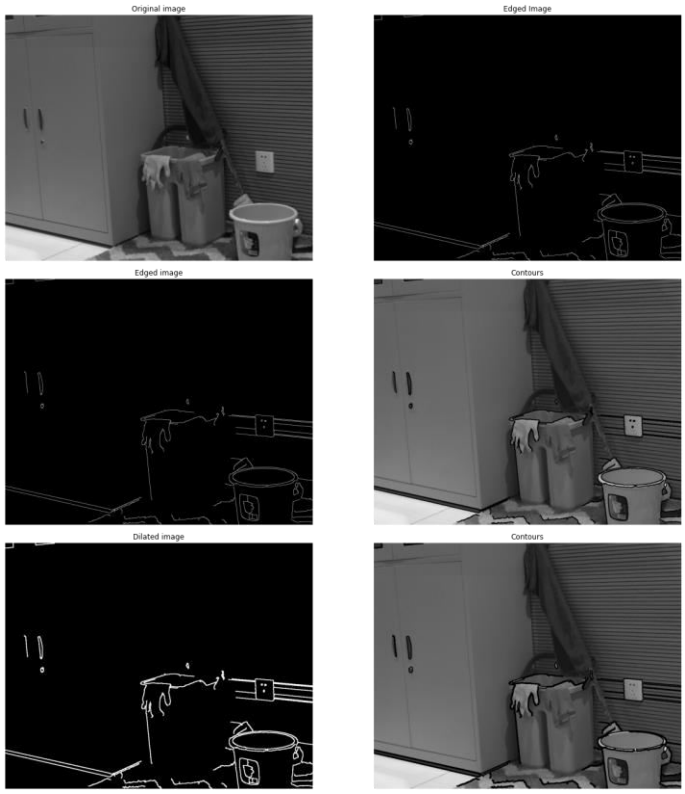
8.4 Closing

Closing is a process of removing small holes from the foreground of an image and placing them in the background. This process, which consists of erosion followed by dilation, modifies the image.

$$(A \bullet B) = (A \oplus B) \ominus B$$

9. Contour Detection

A contour is a closed curve joining all the continuous points having some color or intensity, they represent the shapes of objects found in an image. Contour detection is a useful technique for shape analysis and object detection and recognition.

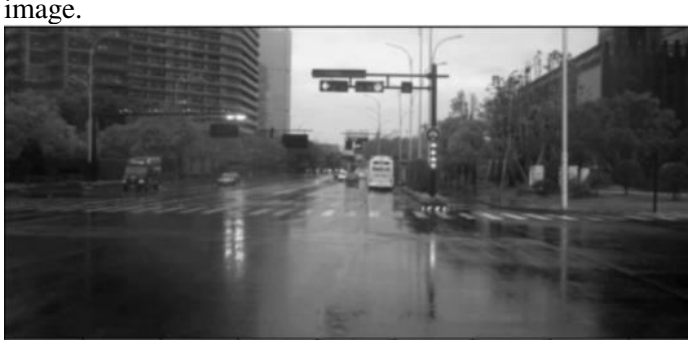


Analysis

Contour detection on a smoothened image and a dilated image was performed. Dilation is done to add some pixels and increase the foreground objects. This will allow the contour detection algorithm to detect only the boundaries of the objects in the image. It can be observed that the number of contours obtained in smoothened image is more when compared to that of the dilated image. The conclusion from the above analysis is that contour detection after dilating an image is a better option to obtain an optimal solution.

10. Morphological Top Hat Operator

A simple morphological image processing technique used to extract the small and fine details from an image.
It consist of two steps :
(i) Obtain the 'opening' of the image.
(ii) Subtract the opening of image from the original image.



(a) Source Binary Image (b) Result of top hat operator

Analysis:

We can use the top hat transform to retrieve small and fine details from an image..

11. Clustering

K-means clustering is a technique for partitioning a set of data points into K clusters, where each data point belongs to the cluster with the nearest mean. In image processing, K-means clustering can be used to classify the pixels in an image into K different clusters, based on the pixel values (e.g., intensity values).

K-means clustering is an iterative process that aims to minimize the sum of the distances between each pixel and its assigned centroid. It is a simple and fast technique that is widely used in image processing and computer vision.

One advantage of K-means clustering is that it can be used to redu dimensionality of an image by compressing it into K clusters. Th be useful for tasks such as image compression, image segmentatio feature extraction.

(a) Input Image (b)Segmented Image using K Means Clustering

Analysis:

The algorithm was able to segment the areas of commonality in the image and hence compressing and segmenting it together. The road appears in one shade distinct from other objects around it and is hence helpful for this project.

12. Hit Or Miss Transform (Shape Detection)

An morphological compound transformation that can be used in finding a given configuration / pattern in a binary image. This technique uses basic morphological image processing as a basis (i.e erosion and dilation).

This morphological transform finds those pixels whose neighborhood matches the shape of a first structuring element B1 while *not* matching the shape of a second structuring element B2 simultaneously.

$\square(*)\square = ((\square \ominus \square 1) \cap (\sim \square \ominus \square 2))$.

The complement of the image can be obtained by simply applying the inverse transform of the thresholded image.

We here try to detect other vehicles (such as the two wheeler in the foreground of the image) that is in close proximity to the camera system. For this, we choose our structuring element B1 to be a rectangle of ones (with dimensions roughly equal to the shape of the two wheeler)

(a) Source image (depth map of scene) (b) Erosion of image by structural element B1 (rectangle of ones) © Erosion of image by structural element B2 (rectangle of zeros, with ones in the middle column)

The single white line gives us the position of the matched object, which in this case is the driver.

Analysis:

The hit or miss transform works extremely well in segmenting / retrieving objects of particular shape in our image, when parts of the image are not sticking to each other (I.e objects of similar shape are not overlapping with each other). To resolve this, we must perform watershed algorithm or erosion as a preprocessing step prior to performing the hit or miss transform.

13. Depth / Disparity Map Generation

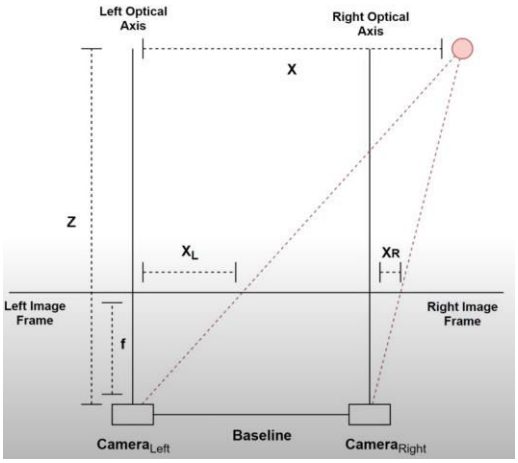
When we take a photo on a camera, we 'convert' the scene's world coordinate into the view / camera coordinate frame, where the camera is at the center, looking down on either the +z or -z axis. Then, we apply the projection matrix and perspective divide to go from the view space onto pixel space.

Our goal is the estimate the relative objects in the image, given camera calibration parameters and a pair of stereo (i.e left and right images).

Essentially, we try to estimate the world coordinates of different pixels in the image.

Stereo Vision System

It consists of two cameras located at a known distance from each other, at the time the pictures of the scene are taken. Often, they have a difference in their horizontal displacement, which is the basis for depth estimation. We need to calibrate the cameras so as to accurately determine the correspondence between pixels in the left and right stereo images. This often involved a template / pattern matching algorithm.



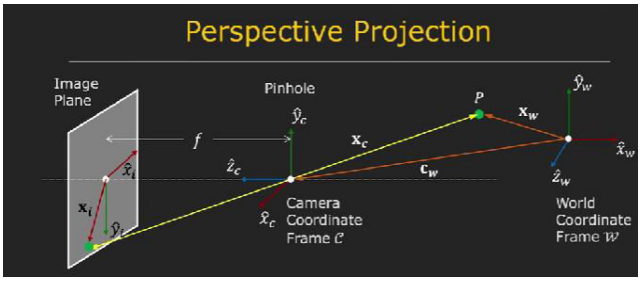
(a) Basic visualization of stereo camera setup

14. Camera Calibration and Parameters

View matrix : Defined using the position and orientation of the camera with respect to the world coordinate frame. The view matrix translates and rotates the scene such that the camera is placed at the origin of the view space, looking down at the z axis (+ve or -ve)

This is referred to the external parameters of the camera

Projection Matrix : Maps point from the view space to the projection plane, where the distance between the camera's optical center and the projection / imaging plane is the focal length of the camera, f. These are referred to as internal parameters of the camera (such as focal length).



Suppose we had a point in the view space, V(xv, yv, zv) and its corresponding point in the image plane, IP(x, y). The relationship between both are given by :

$$\square = \square * \square\square / \square\square$$

$$\square = \square * \square\square / \square\square$$

However, the image sensor has pixels, and the image place's pixel need not have a 1-1 relation with the image sensor pixels.

We can approximate image pixels and the image sensor pixels using the following logic:

If dx and dy are the pixel densities in a millimeter range around a pixel (i.e it has the dimensions px/mm), then the pixel coordinates in the imaging plane is given by,

$$\square_{\square\square\square\square\square\square} = \square\square * \square * \square\square / \square\square$$

$\square_{\square\square\square\square\square\square}$ = dy * f * yv / zv

Also, the origin for the image lies in one corner of the sensor, given by ox and oy. So, finally, the pixel coordinates (u, v) of the sensor w.r.t the view coordinate space pixel coordinates is given by :

$$(\square, \square) = (\square\square * \square * \square\square / \square\square + \square\square, \square\square * \square * \square\square / \square\square + \square\square)$$

It is to be noted that the intrinsic parameters of the camera, that represent the camera's internal geometry is represented by (fx, fy, ox, oy), where fx and fy are the result of

We can convert this into a matrix form which yields:(

(u, v, 1) = [fx 0 ox 0] * [xc yc zc 1]^T

[0 fy oy 0]

[0 0 0 0]

[0 0 0 1]

This is referred to as the external parameters of the camera.

Analysis

Given a projection matrix, we can extract the camera's internal geometry details (i.e the base length and focus x and focus y), as this data is present in ProjectionMatrix[0,0], [1, 1] and the transformation vectors for right and left camera configuration.

15. Disparity Calculation using Template Matching

Disparity map refers to the apparent pixel difference or motion between a pair of stereo images.

Since the displacement in the vertical y axis is 0, we can have a moving window in scanline order (i.e x axis, going from left to right). In the left stereo image, we place a template : usually of a rectangle and try to find the template's corresponding position in the right stereo image. The horizontal position (x) would vary, while the vertical coordinate (y) remains same as the templates y coordinate. We will use the Sum of absolute differences metric to determine the values of ul and ur in the above explanatory image.

$$\square\square\square = \sum_{\square=0}^{\square} \sum_{\square=0}^{\square} (\square\square\square(\square\square\square\square\square_{\square\square\square h\square}[\square,\square] - \square\square\square\square\square_{\square\square\square\square}[\square,\square]))$$

'K' is the template size. The position on the right image which corresponds to minimum SAD value is chosen to ur. The difference of ur and ul is the disparity.

(a) Template (rectangle) positioned on left image (top left of image, on the billboard) (b) Matched template on right stereo images, matched using Sum of Absolute Difference (SAD) metric. Result comes out to 12.

(a) Source image (b) Computed disparity map with block size 5.

\Analysis:

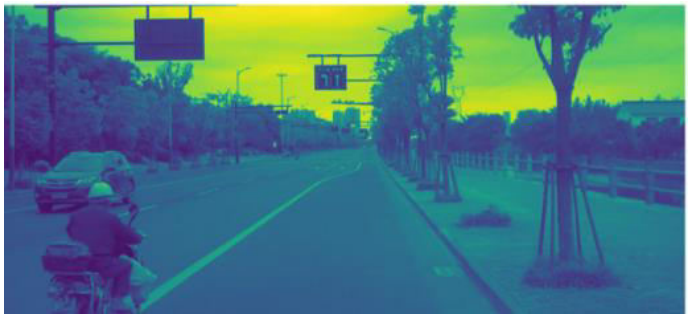
This method of pattern / template matching does not fair well when the image is composed of heavily repeating textures. Alternative adaptive methods of template matching should be considered for this technique (especially if relative camera position data is not available in the camera calibration details).

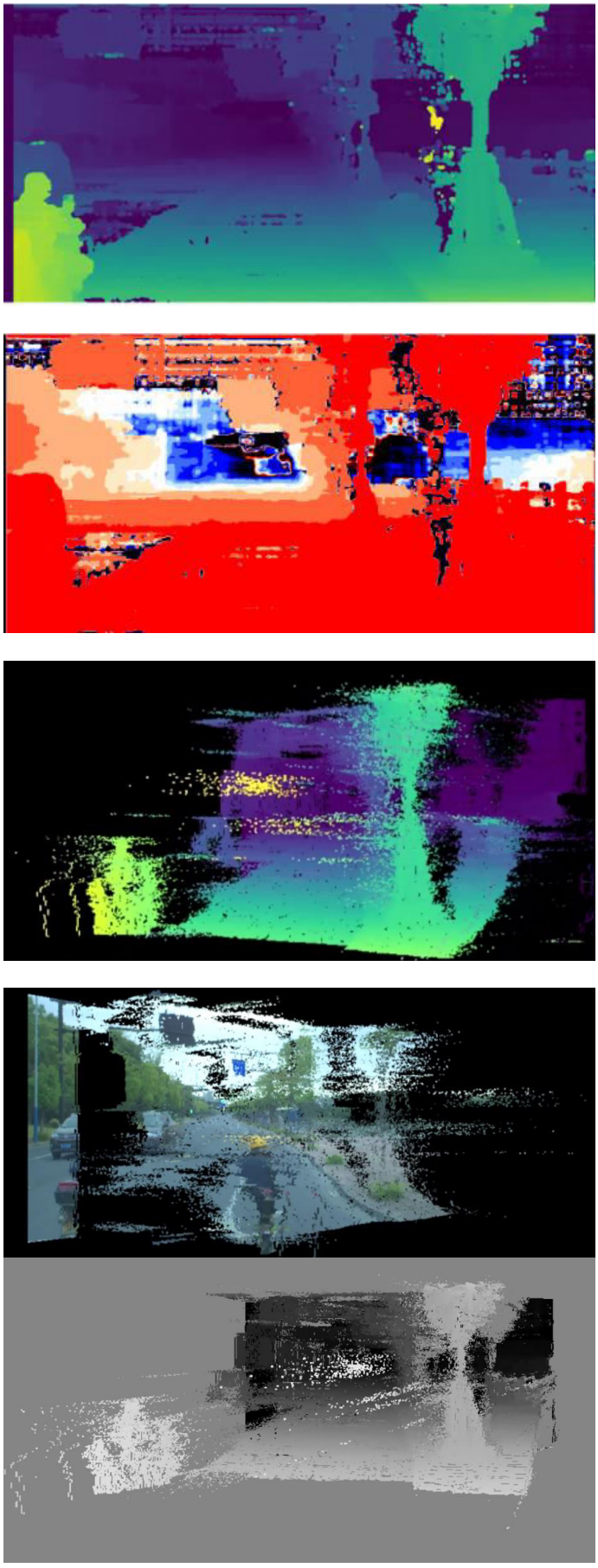
16. Depth Map Calculation

For Generation of Depth map, we need to use the internal camera geometry parameters (ie. baseline and focal length) and disparity map as follows:

$$\square = \square * \square(\square\square - \square\square) = \square\square / \square$$

Here, f is the focal length (obtainable from the camera matrix, b is the baseline values corresponding from the translation vectors t, d is the disparity map.





(a) Source Image (b) Disparity map © Depth map (d,e,f) Point cloud visualization using depth from depth map, and point colors as corresponding disparity map, source image and depth map.

Analysis

By computed the depth information from a given set of stereo images, we can visualize the objects in the scene captured by the image in a 3D coordinate system, by using a point cloud visualization (as displayed in images (d-e). In the case of autonomous vehicles, we can use the depth information to enable the vehicle to take corrective measures (such as halting the vehicle if an object is in close proximity to the vehicle). Also, given an arbitrary number of images and camera calibration details, we can perform full 3D reconstruction of the scene captured by the camera. This can be used in 3D modeling scenarios or advanced visualization.