2-1-2007

Quantitative analysis of infrared contrast enhancement algorithms

Seth Weith-Glushko

QUANTITATIVE ANALYSIS OF INFRARED CONTRAST ENHANCEMENT ALGORITHMS

Seth A. Weith-Glushko

Bachelor of Science, Rochester Institute of Technology, 2004

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the Chester F. Carlson Center for Imaging Science of the College of Science Rochester Institute of Technology

February, 2007

Signature of Author: ____

Accepted By: ____

Coordinator, M.S. Degree Program

CHESTER F. CARLSON

CENTER FOR IMAGING SCIENCE

COLLEGE OF SCIENCE

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S. DEGREE THESIS

The M.S. Degree Thesis of Seth A. Weith-Glushko has been examined and approved by the thesis committee as satisfactory for the thesis requirement for the Master of Science degree.

____

Dr. Carl Salvaggio

____

Dr. Maria Helguera

____

Robert H. Murphy

____

Date

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The author wishes to thank numerous people. First, I would like to thank the members of my thesis committee: Dr. Carl Salvaggio, Dr. Maria Helguera, and Bob Murphy for their patience and their guidance. I would also like to thank other members of the RIT community, specifically Dr. Jeff Pelz, Dr. Roxanne Canosa, Dr. Bob Kremens, and Dr. Andy Herbert for their time and their knowledge. Moreover, I would like to thank BAE Systems, especially Steve Jamison and Peter Norton, for making this research possible. Likewise, I would like to thank the Rochester Fire Department, namely Executive Deputy Chief John Caufield and the members of Rescue 11 and Ladder 17 for their participation. Last but not least, I would like to thank my family and friends for their support.

ABSTRACT

QUANTITATIVE ANALYSIS OF INFRARED
CONTRAST ENHANCEMENT ALGORITHMS

This thesis examines a quantitative analysis of infrared contrast enhancement algorithms found in literature and developed by the author. Four algorithms were studied, three of which were found in literature and one developed by the author: tail-less plateau equalization (TPE), adaptive plateau equalization (APE), the method according to Aare Mallo (MEAM), and infrared multi-scale retinex (IMSR). Engineering code was developed for each algorithm. From this engineering code, a rate of growth analysis was conducted to determine each algorithm's computational load. From the analysis, it was found that all algorithms with the exception of IMSR have a desirable linear nature.

Once the rate of growth analysis was complete, sample infrared imagery was collected. Three scenes were collected for experimentation: a low-to-high thermal variation scene, a low-to-mid thermal variation scene, and a natural scene. After collecting sample imagery and processing it with the engineering code, a paired comparison psychophysical trial was executed using local firefighters, common users of the infrared imaging system. From this trial, two metrics were formed: an average rank and an interval scale. From analysis of both metrics plus an analysis of the rate of growth, MEAM was declared to be the best algorithm overall.

Chapter 1

INTRODUCTION

Remote sensing is defined as “the field of study associated with extracting information about an object without coming into physical contact with it.” [15] Remote sensing is important as a process because it allows users to obtain information about phenomena that would be dangerous or impossible for them to detect solely with their senses. The process can be modeled as a chain, as seen in Figure 1.

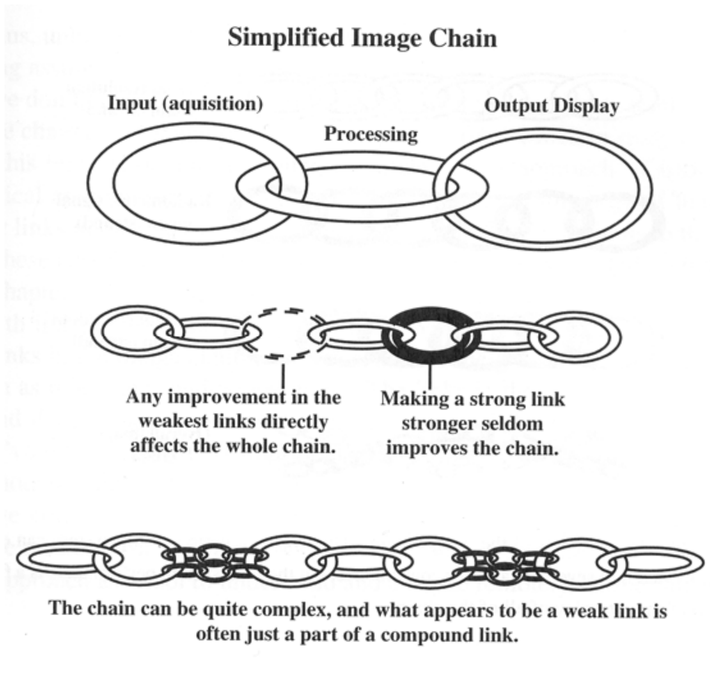

Simplified Image Chain
Input (aquisition)
Processing
Output Display
Any improvement in the weakest links directly affects the whole chain.
Making a strong link stronger seldom improves the chain.
The chain can be quite complex, and what appears to be a weak link is often just a part of a compound link.

Figure 1 – The image chain analogy. Courtesy [15]

In this model, each segment of an imaging system is broken into individual chains: the input link, the processing link, and the display link. A remote senser’s task is to understand how each link in the chain fits together and to avert any problems arising from the interaction between each link. One common problem is data reduction. For example, a detector (input) is able to output pixels that have a dynamic range described by twelve bits. In the same system, the monochrome display (output) is able to output pixels that have a dynamic range of only eight bits. Hence, a procedure aimed at reducing the data must take place in the processing stage to enable the display to work with data from the detector. This procedure must accomplish two goals: reduce the dynamic range of the input image into an image that is acceptable for input by the output system and do this in such a manner that the output image is pleasing to the human observer.

One such procedure is simply changing the hardware in the imaging system; one can use a detector with a lower dynamic range or a display with a higher dynamic range. However, this introduces the opportunity for capturing imagery which is not robust enough for a user's purpose and high costs in developing the system, respectively. Another procedure that can be employed is dynamic range compression. Dynamic range compression can be defined as the mapping of pixels containing a high dynamic range to pixels that contains a reduced dynamic range. In essence, dynamic range compression is a pixel operator, defining the value of an arbitrarily located pixel in a new image by using the value of the corresponding pixel in the original image. Dynamic range compression has a number of applications in fields such as video telephony [1], radiology [7, 16], and high dynamic range photography [8, 14]. As such, research has been performed with the aim of providing a dynamic range compression algorithm that suits an application area's needs, such as enhanced image quality or heightened information availability. Although research has been performed, each study has shown a lack of quantitative metrics to describe how well each algorithm performs in terms of image quality. In most cases, all that is offered is a simple qualitative metric with no explanation of meaning or background.

Another task remote sensers must occupy themselves with is systems integration, or the meshing of input systems, processing systems, and output systems. Systems integration requires that the remote senser define certain parameters of a system and understand how each affects the interplay between each link in the imaging chain. One parameter that is important is that of power consumption. In many commercial industries, power consumption plays a huge role in the development of a system because no consumer will use an imaging system that is rated to last for a few minutes when a competing imaging system can be used for hours. In many imaging systems, the use of digital image processing microprocessors have become prevalent due to their scant size and ability to upload computer programs for real-time processing of imagery from a detector. However, the use of the solid-state image processors is done with care as they are a large source of power consumption concerns in modern imaging systems. This is due to the direct correlation between the need for clock-speed required to apply image processing algorithms in real-time and power usage of the solid-state chip. Unfortunately, no research has been performed that measures each algorithm's processing time requirements using a quantitative metric.

Chapter 2

SPECIFIC AIMS

The specific aims of this research were to:

1. Research and develop algorithms that could perform dynamic range compression and contrast enhancement simultaneously on infrared imagery.
2. Collect sample infrared imagery that would fully test an algorithm's response.
3. Implement engineering code that showed each algorithm's feasibility on example imagery using a simple graphical user interface.
4. Execute an analysis that determined the rate of growth and calculate an estimation of the number of operations required to complete each algorithm on an arbitrarily sized image.
5. Generate video streams from collected imagery to simulate actual camera operation and use them in a paired-comparison psychophysical trial.
6. Run the psychophysical trial to fully determine the algorithms' quality.

Chapter 3

BACKGROUND

To fully understand how the infrared contrast enhancement algorithms will be evaluated, one must first understand how image processing algorithms can be tested. The first way to analyze an algorithm has its roots in computer science. The second way to analyze an algorithm has its roots in psychophysics. By using both methods, a better evaluation of the "best" infrared contrast enhancement algorithm can be determined.

3.1 Algorithm Analysis

In computer science, the process of algorithmic analysis is incredibly important. By finding a quantitative metric of an algorithm's efficiency, decisions involving the algorithm's use in a system can be made; for example, whether an algorithm will execute correctly on a microprocessor system or if further optimization needs to occur. At first glance, the time an algorithm requires to execute might seem to be an appropriate metric. However, the amount of time an algorithm requires is not useful in an algorithmic analysis for two reasons. First, one should be concerned with the relative efficiency of how an algorithm solves a problem. Second, an algorithm does not get "better" or "worse" when transferred to faster or slower computing systems. [9]

As such, computer scientists have determined a way to compare two algorithms through the use of computing resources as a function of input image size. This is done by comparing the rate at which their use of resources grows. The growth rate is critical because there are instances where one algorithm may take fewer operations than another when the input image is small but many more when the image is large. This method is called “Big O” notation analysis. [9]

Specifically, one wishes to find the rate of growth that is asymptotically bound to some function *f*. By finding this “worst-case” bound, one could compare the “worst-case” performance of different algorithms that solve the same problem. If one algorithm has a much larger rate of growth than another, then that algorithm would not be as efficient and hence, would be undesirable.

To find the rate of growth of an algorithm, one must simply find the amount of consumption of a computing resource versus the size of the input. For the analysis of image processing algorithms, the simplest way to accomplish this is to record the amount of time required to execute each algorithm with arbitrarily sized input imagery. Next, a plot of time versus input size is generated. From this plot, an equation is developed to approximate the data. Based on this approximation, the largest term will be determined. This will be the order of the equation.

3.2 Paired Comparison Psychophysical Testing

According to the American Heritage dictionary, psychophysics is defined as the branch of psychology that deals with the relationships between physical stimuli and sensory response. In addition, psychophysics concerns itself with the quantitative measurement of the relationships; in essence, using a human being as a yard stick. Common knowledge dictates that a human being makes for a poor measurement device.

However, with careful thought and planning, a person can be used as an accurate tool. Hence, by performing psychophysical trials, values can be measured that indicate the level of quality for each of the algorithms to be tested in this study.

One type of psychophysical trial is the paired-comparison test. Using the paired-comparison method allows for the generation of an interval scale, a rating of which algorithm is the best. By having an interval scale, a quantitative determination can be made as to the relative quality differences between each of the algorithms. The paired-comparison method asks a human subject to select from two samples which best answers a question put forth to them. The human subject will then compare all possible combinations. By recording which element of each pair the subject selects, certain assumptions can be made that leads to the creation of an interval scale [3]. When testing image processing algorithms, the samples would represent imagery output from the algorithms studied.

To get the best sense of an algorithm's quality, the paired-comparison test can be run on multiple scenes. By testing an algorithm's response for various scenes, a more complete picture of the algorithm's effectiveness can be made. As such, a user will be required to make a certain number of comparisons. This number can be calculated through the use of Equation 1. n_p represents the number of pairs, A represents the number of algorithms while M represents the number of images.

$$n_p = M\frac{A(A-1)}{2} \tag{1}$$

After psychophysical data collection, two informational items can be developed: an average rank and an interval scale. The average rank will be calculated through the use of Equation 2. *Rank*$_{av}$ represents a 1 by n-element vector containing the average rank of the samples. The row vector of ones is n elements long. *F* represents an n-unit square data matrix where each cell indicates how many times one sample in a pair was picked over another. If during a psychophysical experiment an observer selects sample j over i, the value in the cell located at the jth column and the ith row of *F* increases by 1. This continues for each observer and for all pairs. *N* represents the number of observers.

$$Rank_{av} = \frac{1}{N}[1 \quad 1 \quad 1 \quad \ldots \quad 1]F \tag{2}$$

Since the number of observers for the test will not equal the number of people in the testable population (i.e. the human race), there is some error associated with the calculation of the average rank. This error can manifest itself in equal ranks for the same sample. As such, one can use statistics to test whether a rank for one sample is actually the same as another. To do so, one can use a statistical hypothesis test. In a statistical hypothesis test, two statements are formed: a null hypothesis (H_0) and an alternative hypothesis (H_a). In addition, a value called a test statistic is formed. A test statistic is a value on which the decision to reject H_0 is based. Moreover, there exists a rejection region, or the set of all test statistic values for which H_0 will be rejected. By comparing the test statistic to the rejection region, one can decide whether to reject the null hypothesis and accept the alternative hypothesis [2].

To statistically evaluate the average rank, one must first realize that one can generate *N* average ranks by calculating an average rank for each observer. As such, a mean average rank and standard deviation can be calculated for each sample. One must then consider whether the average ranks among the users are different. Hence, the statistical hypothesis that the mean average rank of each sample are different must be tested. To start, it must be assumed that the mean average rank amongst each possible pair of samples are the same. Therefore, the null hypothesis, alternative hypothesis, test statistic, and rejection region can be defined as Table 1.

H_0:	$\mu_{R_i} = \mu_{R_j}$
H_a:	$\mu_{R_i} \neq \mu_{R_j}$
Test statistic:	$t = \dfrac{\overline{R}_i - \overline{R}_j}{\sqrt{\dfrac{s_i^2}{N} + \dfrac{s_j^2}{N}}}$
Reject H_0 if:	$t \geq T_{\alpha/2, \nu}$

Table 1 – Average rank test hypothesis

μ_{R_i} and μ_{R_i} represent the true rank for samples i and j, $\overline{R}_i$ and $\overline{R}_j$ represent the calculated mean average rank for samples i and j, s_i and s_j represent the standard deviation of the average ranks for samples i and j, *t* represents the test statistic, α represents an arbitrary significance level, and $T_{\alpha/2, \nu}$ represents the value of the Student's T distribution at significance $\alpha/2$ and degrees of freedom ν, which can be calculated using Equation 3. The value for the Student's T distribution can be found by using standard tables. The Student's T distribution was selected because the amount of trial participants is not expected to be enough to assume a Gaussian nature and thus, the use of the normal distribution.

$$\nu = \frac{\left(\frac{s_i^2}{N} + \frac{s_j^2}{N}\right)^2}{\frac{(s_i^2 / N)^2}{N-1} + \frac{(s_j^2 / N)^2}{N-1}} \tag{3}$$

To start, the unique combination of means and standard deviations that can be made from the samples is determined. Second, the test statistic and rejection region are calculated. Once done, the test statistic is compared to the rejection region. If the test statistic is greater than the specified *t* value, the null hypothesis is rejected and one can safely assume that the true rank value is different from the data. Once this test has been performed on each pair and the values are compared, one can see that it is desirable to have the test hypothesis rejected in favor of the alternative hypothesis. Put another way, by having average ranks that are statistically separable (i.e. not the same), one can make an accurate determination of the true rank. This is important because one can then infer that the interval scales will be different.

Another way to visualize the error inherent in the calculation is through the use of confidence intervals. A confidence interval is a range of numbers where the true value of a statistical parameter may fall with a desired probability. To calculate a confidence interval, one may use Equation 4.

$$CI_i \in [\overline{R}_i - T_{1-\alpha/2, N-1}\frac{s_i}{\sqrt{N}}, \overline{R}_i + T_{1-\alpha/2, N-1}\frac{s_i}{\sqrt{N}}] \tag{4}$$

CI_i represents the confidence interval for sample i while $T_{1-\alpha/2,N-1}$ represents the inverse cumulative distribution function of the Student's T distribution at a 1-α/2 critical value and N-1 degrees of freedom.

After the average ranks are calculated, one can generate an interval scale. To calculate an interval scale, Thurstone's Law of Comparative Judgment will be used. The law states that for various reasons, an observer might vary his response for the same sample pair. This variance is assumed to have a Gaussian distribution. Based on this law, a number of assumptions and steps can be taken to generate a scale. First, Thurstone found that the proportion of times that a sample was chosen over another is an indirect measure of the distance between the two on an interval scale. Accordingly, one can generate a matrix that contains these proportions using Equation 5. *P* represents the proportionality matrix.

$$P = \frac{1}{N}F \tag{5}$$

Next, one can back out each of the values in the proportionality matrix as differences in scale through the use of Equation 6. *P(A>B)* represents a cell within the proportionality matrix, H^1 represents the inverse of the Gaussian cumulative distribution function, and S_A-S_B represents the scale difference.

$$S_A - S_B = H^{-1}[P(A > B)] \tag{6}$$

Upon using Equation 5, we then create a matrix *S* that contains each of the scale differences, as seen in Equation 7.

$$S = \begin{bmatrix} S_1 - S_1 & S_2 - S_1 & \cdots & S_n - S_1 \\ S_1 - S_2 & S_2 - S_2 & \cdots & S_n - S_2 \\ S_1 - S_3 & S_2 - S_3 & \cdots & S_n - S_3 \\ \vdots & \vdots & \ddots & \vdots \\ S_1 - S_n & S_2 - S_n & \cdots & S_n - S_n \end{bmatrix} \tag{7}$$

One should note that the sum of each column reduces to the scale represented by that column by the average of all scales. This can be seen mathematically for the first column as Equation 8.

$$\frac{1}{A}\sum_{i=1}^{A}(S_1 - S_i) = S_1 - \overline{S} \tag{8}$$

Hence, by setting an arbitrary scaling such that the average of the scales is zero, each column sum will return the scale value for that sample [3].

Since the number of observers for the psychophysical trial will not be the same as the number of observers in the entire human population, there is some error associated with each scale value. Montag defines the error interval as Equation 9.

$$\Delta S_{U|L} = S_i \pm z_{1-\frac{\alpha}{2}} \sigma_{\bar{x}} \tag{9}$$

$\Delta S_{U|L}$ represents the upper and lower error bounds, S_i represents an arbitrary scale value, z represents the z-score specified by the cutoff formulated from α, and $\sigma_{\bar{x}}$ represents the standard error. Montag further defines the standard error as Equation 10 [10].

$$\sigma_{\bar{x}} = 1.76(A + 3.08)^{-0.613}(N - 2.55)^{-0.491} \tag{10}$$

Once the error interval has been found, it can be used to determine whether the scale values are statistically different by forming error bars in a plot of interval scale versus algorithm. The scale values are deemed statistically different if "the error bar of one does not extend past the mean of another." [11]

Chapter 4

ALGORITHMS

At the heart of this research is the development of the infrared contrast enhancement algorithms. In general, infrared contrast enhancement algorithms are designed to take high dynamic range input infrared imagery and output low dynamic range display imagery. As such, the input image will be called *f(x,y)* and the output image will be called *g(x,y)* for the purpose of algorithmic explanation.

In all, there are four algorithms to be studied: tail-less plateau equalization (TPE), adaptive plateau equalization (APE), the method according to Aare Mällo (MEAM), and infrared multi-scale retinex (IMSR). It should be noted that there are two additional algorithms to be explained: histogram equalization (HE) and linear scaling (LS). These algorithms are components of the four algorithms and are not tested independently.

4.1 Histogram Equalization (HE)

Histogram equalization is a way of increasing the amount of entropy in an image by re-mapping the values in the pixels of an image such that there is an equal chance of each grey level appearing within an image. In terms of automatic dynamic range compression and contrast enhancement, histogram processing provides a method to map values from the high dynamic range image into a lower dynamic range image such that the contrast in the image is enhanced based on the probability of a certain digital count appearing in the image.

To start, a histogram of the input image *f(x,y)* is taken. A histogram is simply a discrete function *H(k)* where *k* is a grey level within the image and *H(k)* is the number of pixels with the specified grey level *k* within said image. From this histogram, the probability density function (PDF) and the cumulative distribution function (CDF) are found through Equations 11 and 12.

$$PDF(k) = \frac{H(k)}{N} \text{ (for all } k \in f(x,y)) \tag{11}$$

$$CDF(k) = \sum_{a=0}^{k} PDF(a) \tag{12}$$

N represents the total number of pixels within the input image. From the CDF, a mapping of a high dynamic range value to a low dynamic range value can be made through Equation 13.

$$m(k) = \lfloor L_{\max} CDF(k) \rfloor \tag{13}$$

m(k) is the output digital count that is mapped to the input digital count *k* while L_{max} is the highest digital count possible in the lowered dynamic range system [5]. This mapping often comes in the form of a lookup table. A lookup table is an entity which has two columns: one for the input greyscale value and one for the corresponding output greyscale value. To perform histogram equalization, one must apply the lookup table generated by Equation 13 to the input image *f(x,y)*. This is done pixel-by-pixel, finding the appropriate mapping for each input pixel to generate each output pixel. Although histogram equalization does increase the entropy and balances an image's histogram, the process has the undesirable effect of removing detail from highlight and shadow areas within the image. Subjectively speaking, this effect tends to make the image look artificial and thus, uninformative to a human observer. An example of this effect can be seen in Figure 2.

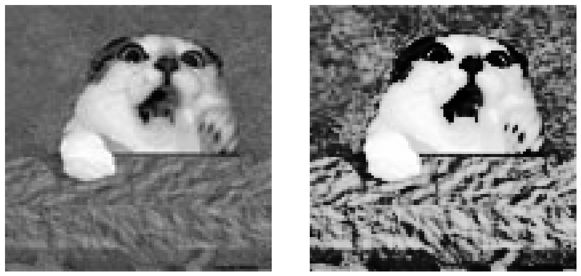


Figure 2 – Loss of detail in mouth region due to histogram equalization

4.2 Linear Scaling (LS)

In linear scaling, an input infrared image is transformed into a compressed output image through the use of two linear functions. An overview of linear scaling can be seen in Figure 3.

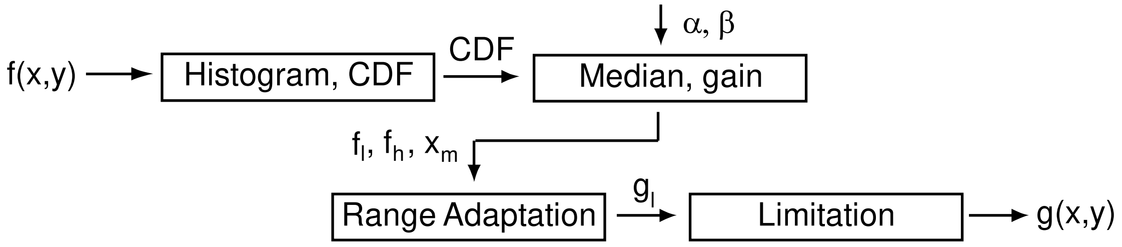


Figure 3 – Linear scaling

The histogram and corresponding CDF of the input image *f(x,y)* are found using Equations 11 and 12. To keep stray pixels from making the gain unnecessarily small, a percentage α of the histogram is removed from each end of the distribution. As such, α can have values from zero to one-half. Using the CDF, three values are defined. x_a is equal to the value k

which satisfies the equation *CDF(k)* = α. x_b is equal to the value *k* which satisfies the equation *CDF(k) = 1 –* α. x_m is the median value $_k$ which satisfies the equation *CDF(k) =* ½. After finding these values, the dynamic range of the output is found and the corresponding linear gains that need to be applied can be calculated using Equations 14 and 15. Figure 4 depicts this process visually.

$$y_a = y_{\max}\beta + y_{\min}(1-\beta), y_b = 1 - y_a, y_m = \frac{y_{\max} - y_{\min}}{2} \qquad (14)$$

$$f_l = \frac{y_m - y_a}{x_m - x_a}, f_h = \frac{y_b - y_m}{x_b - x_m} \qquad (15)$$

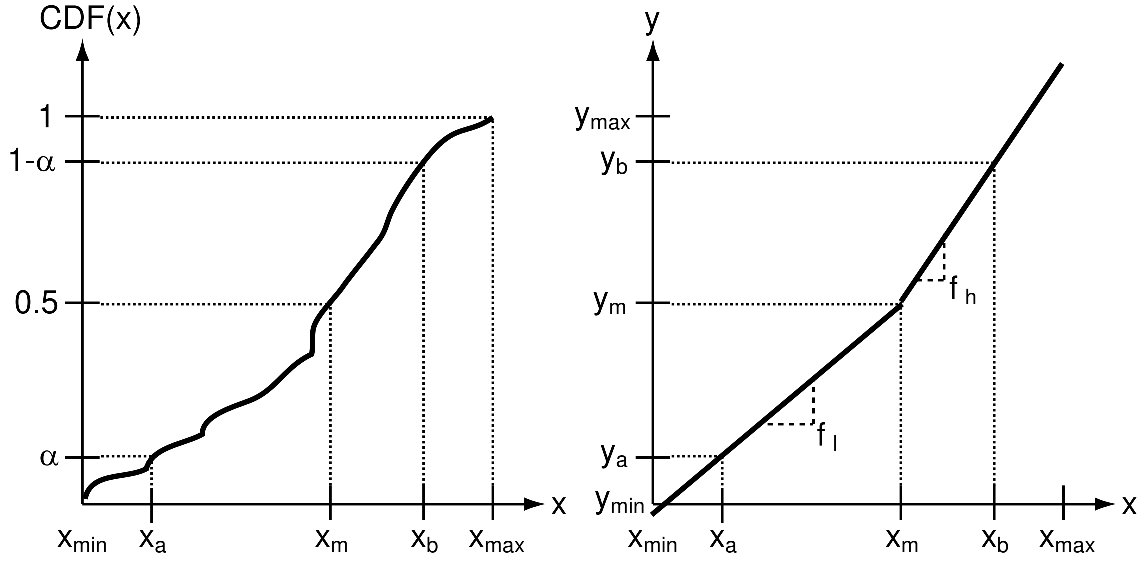


Figure 4 – Finding gains and median for linear scaling

β represents a scalar that scales the input range to be scaled. Traversing the input image, the pixels in the dynamic range adapted image are found using Equation 16.

$$g_l(x,y) = \begin{cases} f_l(f(x,y) - x_m) & f(x,y) \le x_m \\ f_h(f(x,y) - x_m) & f(x,y) > x_m \end{cases} \tag{16}$$

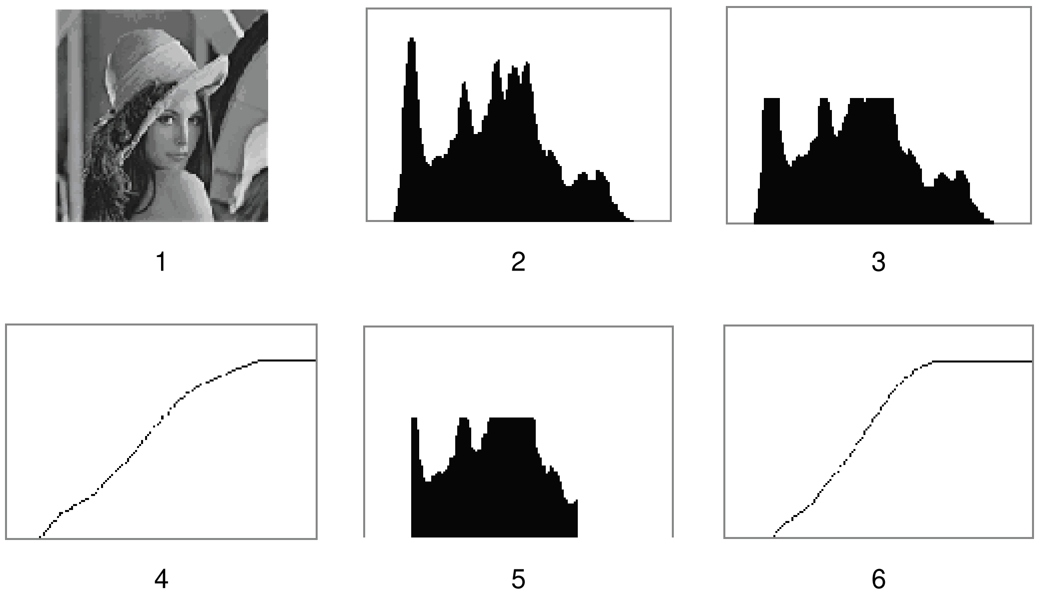
Once this adapted image has been formed, the results are limited to ensure a pixel's digital count is within the range of desired display using Equation 17.

$$g(x,y) = \begin{cases} y_{\min} & g_l(x,y) \le y_{\min} \\ g_l(x,y) & y_{\min} < g_l(x,y) < y_{\max} \\ y_{\max} & g_l(x,y) \ge y_{\max} \end{cases} \tag{17}$$

"Linear scaling is a simple method and image with a large dynamic range lose much of the detail and important high frequency content is reduced too much" [4]

4.3 Tail-less Plateau Equalization (TPE)

Tail-less plateau equalization is a variation on histogram equalization where a maximum gain parameter, called the plateau, is introduced. The plateau is a clipping value that is applied to a histogram, placing a limit on the number of pixels that can be resident within each histogram bin. The purpose of the plateau is to lessen the chance for excessive contrast enhancement. It does so by making the lookup table more linear, increasing the probability that all possible input pixel values will be present in the output image. An overview of the process can be seen in Figure 5.



1. An image is selected
2. The image's histogram is generated
3. The histogram is clipped to a pre-defined parameter
4. A CDF is created from the clipped histogram
5. Using the clipped CDF, the leading and trailing tails of the histogram are zeroed
6. A CDF is created from the tail-less histogram and is used in the mapping

Figure 5 – Tail-less plateau equalization

To start, the histogram of the input image *f(x,y)* is calculated, which from this point on will be referred to as *H(k)*. In plateau equalization, a maximum gain parameter (P_{max}) is introduced and a new histogram $H_p(k)$ is calculated, as seen in Equation 18.

$$H_p(k) = \begin{cases} H(k) & H(k) \le P_{\max} \\ P_{\max} & H(k) > p_{\max} \end{cases} \tag{18}$$

Using this modified histogram, the PDF and CDF of $H_p(k)$ is calculated using Equations 11 and 12 [4]. Next, the "tails" of the modified histogram are eliminated using Equation 18, forming a new histogram $H_t(k)$. By removing the tails of the histogram, outlier pixels can be forced into saturation, increasing the contrast in the output image. t_{max} is a value between zero and one-half and represents the percentage of pixels one wishes to remove from the head and tail-end of the histogram. $CDF_p(k)$ represents the CDF of $H_p(k)$.

$$H_t(k) = \begin{cases} H_p(k) & CDF_p(k) \in [t_{\max}, 1 - t_{\max}] \\ 0 & otherwise \end{cases} \tag{19}$$

The PDF and CDF of the new histogram $H_t(k)$ are calculated and Equation 13 is used to generate a lookup table, which is applied to each pixel globally to form the output image $g(x,y)$.

4.4 Adaptive Plateau Equalization (APE)

Adaptive plateau equalization is very similar to tail-less plateau equalization in the sense that a plateau is applied to a histogram before a calculation of the mapping function is determined. However, instead of having the maximum gain parameter fixed, it is adapted to the current histogram of the scene. First, the histogram $H(k)$ is taken of the scene and its corresponding CDF $CDF(k)$ is calculated using Equations 11 and 12. Second, a number of values are defined. These values are illustrated in Figure 6 and a description of these values is provided in Table 2.

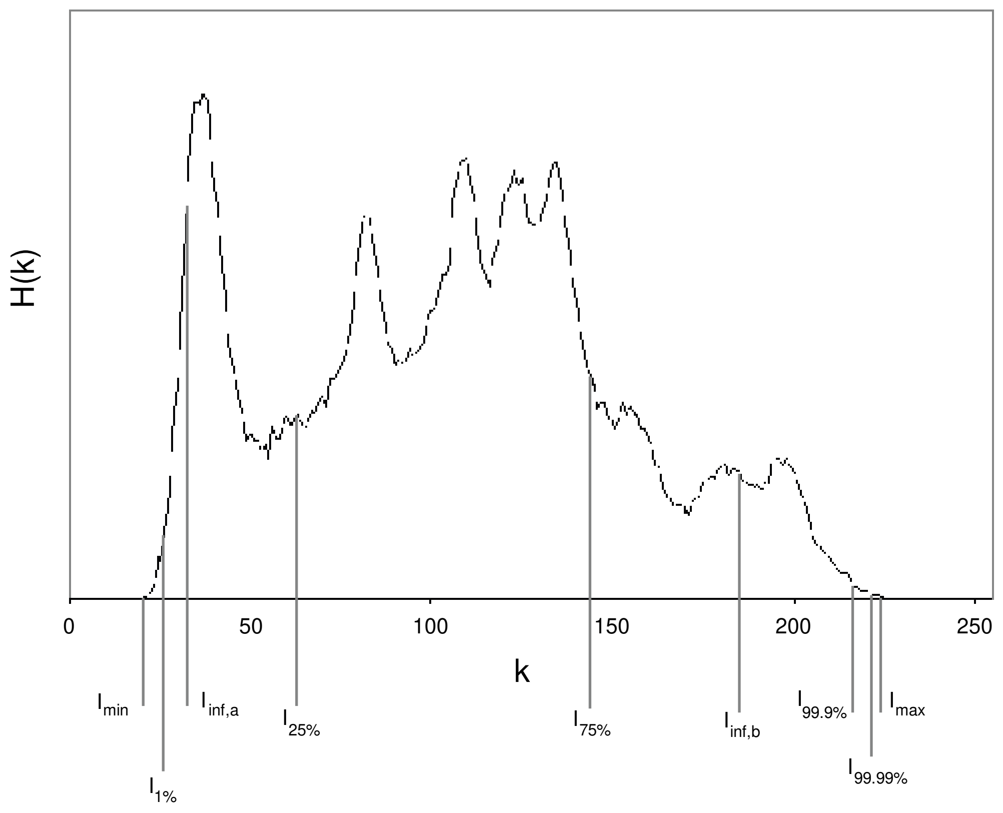


Figure 6 – Parameters necessary for adaptive plateau equalization

Value	Description	Found By
I_{min}	The greyscale that corresponds to the first histogram bin with a value greater than zero	The first greyscale k where CDF(k) > 0
I_{max}	The greyscale that corresponds to the last histogram bin with a value greater than zero	The last greyscale k where CDF(k) > 0
$I_{1\%}$	The greyscale that corresponds to the location in the CDF that is equal to .01	The greyscale k that satisfies CDF(k) = 0.01
$I_{99.9\%}$	The greyscale that corresponds to the location in the CDF that is equal to .999	The greyscale k that satisfies CDF(k) = 0.999
$I_{99.99\%}$	The greyscale that corresponds to the location in the CDF that is equal to .9999	The greyscale k that satisfies CDF(k) = 0.9999
$I_{25\%}$	The greyscale that corresponds to the location in the CDF that is equal to .25	The greyscale k that satisfies CDF(k) = 0.25
$I_{75\%}$	The greyscale that corresponds to the location in the CDF that is equal to .75	The greyscale k that satisfies CDF(k) = 0.75
$I_{inf,a}$	The first inflection point of the histogram	See below
$I_{inf,b}$	The last inflection point of the histogram	See below
η_A	The number of pixels with a value less than $I_{inf,a}$	$\eta_A = \sum_{i=I_{\min}}^{I_{\inf,a}-1} H(i)$
η_B	The number of pixels with a value greater than $I_{inf,b}$	$\eta_B = \sum_{i=I_{\inf,b}+1}^{I_{\max}} H(i)$

Table 2 – Important values for adaptive plateau equalization

As one can see from Figure 6 and Table 2, almost all of the values needed for the plateau algorithm can be found using the CDF. It should be noted that the values of the CDF are not likely to match the limits specified in Table 2 exactly. Hence, the greyscale that corresponds to the CDF value that is closest to the desired value will be used. $I_{inf,a}$ and $I_{inf,b}$ are not derived from the CDF but from the shape of the histogram. These two inflection points are found by applying a moving window sum of width w across the histogram. The low ($I_{inf,a}$) and high ($I_{inf,b}$) inflection points correspond to intensities where the moving window sums change from their previous values by a threshold amount based on a fraction of the difference between I_{min} and I_{max}, as seen in Equation 20. ΔI represents the threshold amount and ε represents a scalar value that ranges between zero and one.

$$\Delta I = \varepsilon(I_{\max} - I_{\min}) \tag{20}$$

Limitations are placed on the high and low inflection points. These limitations are described in Equations 21 and 22.

$$I_{\inf,a} = \{I_{\inf,a} = k \ni I_{\inf,a} < I_{25\%}\} \tag{21}$$

$$I_{\inf,b} = \{I_{\inf,b} = k \ni (I_{\inf,b} > I_{75\%}) \,\&\, (I_{\inf,b} < \frac{I_{\max} - I_{\min}}{2})\} \tag{22}$$

After the inflection points have been defined, η_A and η_B can be found by using the appropriate equation in Table 2.

Next, the maximum gain parameter can be calculated. To do so, a number of intermediate values are calculated. The first is the ratio of pixels that occupy that central portion of the histogram versus the tails of the histogram. This ratio is defined as Equation 23. *X* represents the ratio while *N* represents the total number of pixels in the image.

$$X = \frac{N - (\eta_A + \eta_B)}{\eta_A + \eta_B} \tag{23}$$

After the ratio has been calculated, the nominal plateau value can be calculated as P_{nom}.

$$P_{nom} = \begin{cases} \dfrac{X\eta_B}{I_{\inf,b} - I_{\inf,a}} & I_{99.99\%} - I_{\inf,b} > I_{\inf,a} - I_{1\%} \\ \dfrac{X\eta_A}{I_{\inf,b} - I_{\inf,a}} & I_{99.99\%} - I_{\inf,b} \le I_{\inf,a} - I_{1\%} \end{cases} \tag{24}$$

Next, the dynamic range of the scene (R_D) can be calculated using Equation 25.

$$R_D = I_{99.9\%} - I_{1\%} \tag{25}$$

Using this dynamic range metric, a dynamic range adjustment factor can be calculated using Equation 26. F_{DR} represents the dynamic range adjustment factor while L_{max} represents the maximum grey level output after processing.

$$F_{DR} = \begin{cases} 1 - \dfrac{L_{\max}}{R_D} & R_D > L_{\max} \\ 1 - \dfrac{R_D}{L_{\max}} & R_D \le L_{\max} \end{cases} \tag{26}$$

In addition to the dynamic range adjustment factor, another adjustment factor is calculated to create a more natural appearance of extended dark regions whose intensities are greater than $I_{75\%}$. The adjustment factor F_{ED} is computed as:

$$F_{ED} = 1 - \frac{I_A - I_{0.1\%}}{I_B - I_{0.1\%}} \tag{27}$$

The actual gain parameter (P_A) that will be used to perform plateau equalization can be calculated using Equation 28. A requirement will be placed upon this gain parameter that the value must be greater than or equal to one.

$$P_A = P_{nom} \cdot F_{DR} \cdot F_{ED} \tag{28}$$

Due to the ever-changing nature of infrared imagery, it is possible that the adaptive plateau value can change in value greatly from image to image. To make this algorithm suitable for video, a temporal lowpass infinite impulse response (IIR) filter has been applied to the plateau value. This entails taking a previously calculated plateau value

$P_{A,previous}$ and forming a new plateau value $P_{A,filtered}$ using Equation 29. $P_{A,current}$ represents the plateau value calculated for the current image.

$$P_{A,filtered} = \psi P_{A,current} + (1-\psi)P_{A,previous} \tag{29}$$

where ψ represents a scalar arbitrary set to a value between zero and one. By changing the value of ψ, the amount of a current scene's contribution can be minimized. Hence, flicker can be reduced in the resulting video stream.

Once the gain parameter has been calculated, a new histogram is calculated based on the original histogram [6] where $H_p(k)$ represents the new plateau histogram.

$$H_p(k) = \begin{cases} H(k) & H(k) \le P_F \\ P_A & H(k) > P_F \end{cases} \tag{30}$$

Using Equations 11 and 12, a CDF is calculated using the plateau histogram. Then, using Equation 13, a mapping of pixel values is formed using the CDF to produce the output image *g(x,y)*.

4.5 Method According to Aare Mällo (MEAM)

The method according to Aare Mällo is a departure from performing histogram equalization as the sole dynamic range compression and contrast enhancement instrument. MEAM separates an input image *f(x,y)* into high spatial frequency and low spatial frequency component images. By operating on the two component images separately, a higher degree of control over the dynamic range compression and contrast enhancement can be realized. An overview of the algorithm can be seen in Figure 7.

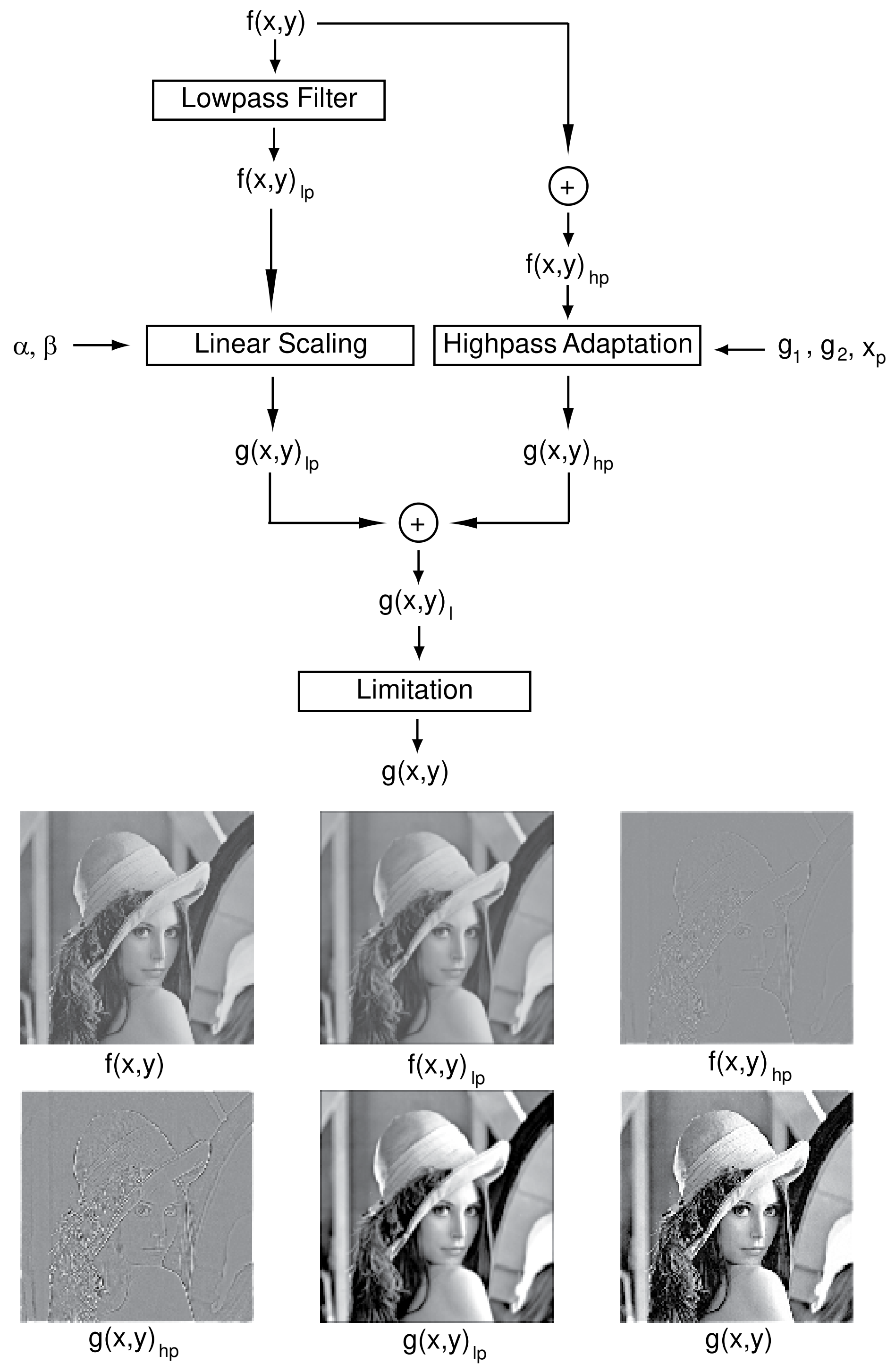


Figure 7 - Overview of MEAM

In this algorithm, the input image *f(x,y)* is convolved with a lowpass filter, forming a low-pass image ($f(x,y)_{lp}$). This filter is often a mean value spatial filter. The original image has the low-pass image subtracted from it forming a high-pass image ($f(x,y)_{hp}$). A gain is applied to the high-pass image. This gain is applied using Equation 31. x_p represents an arbitrary pixel value above which one gain value (g_2) is applied and below which, a different gain value (g_1) is applied. This value is specified such that the edge information pixels, which usually have a small value, are enhanced by the first gain parameter while pixels containing noise information, which usually have a large value, are attenuated by the second gain parameter.

$$g(x,y)_{hp} = \begin{cases} g_1 f(x,y)_{hp} & \left|f(x,y)_{hp}\right| < x_p \\ g_2 f(x,y)_{hp} & \left|f(x,y)_{hp}\right| \geq x_p \end{cases} \tag{31}$$

To the low-pass image, a number of histogram enhancement processes can be performed. For simplicity here, a linear scaling will be performed. The transformed low-pass and high-pass images are summed (forming $g(x,y)_l$) and the resultant values are limited between a specified range, forming the final image *g(x,y)* [6].

4.6 Infrared Multi-Scale Retinex (IMSR)

The infrared multi-scale retinex is an offshoot of a neuro-physiological model called the Retinex calculation. The Retinex calculation "was perceived as a model of the lightness and color perception of human vision." [13] This model was based on the receptive field structures found within the human visual system. By using a specifically designed spatial filter called a surround, the lateral opponent operation of the human visual system could be mimicked. Later research showed that the model could be applied to existing imagery to enhance it in a way that the eye would find aesthetically pleasing. Expounding on this research, Rahman found that by using this model with a Gaussian surround, existing imagery could be improved. Rahman called this process the "multi-scale retinex" and its process is defined mathematically in Equation 32.

$$R(x,y) = L_{\max} \sum_{s=1}^{S} W_s \{\log I(x,y) - \log[F_s(x,y) * I(x,y)]\} \tag{32}$$

R(x,y) represents the output value at location (x,y), *S* represents the number of surround functions one wishes to use to perform the retinex calculation, L_{max} represents the maximum grey value possible in the output image, *I(x,y)* represents the input image rescaled from zero to one at location (x,y) while W_s represents an arbitrary numerical weight associated with the sth Gaussian surround function defined by F_s. F_s is defined as:

$$F_s(x,y) = \kappa e^{\frac{-(x^2+y^2)}{\sigma_s^2}} \tag{33}$$

σ_s^2 represents the variances of the different scales one uses in the calculations. κ represents a numerical weighting that can be calculated as seen in Equation 34.

$$\kappa = \frac{1}{\sum_x \sum_y F(x,y)} \tag{34}$$

Rahman found that using multiple surrounds is necessary to achieve a balance between dynamic range compression and correct tonal rendition. Through experimentation, three scales comprising of a narrow ($\sigma = 5$), medium ($\sigma = 20$) and wide surround ($\sigma = 240$) with equal weightings was sufficient [13].

After applying the multi-scale retinex to infrared imagery, it was found that due to the dynamic nature of infrared imagery, the logarithmic operation would not be sufficient to compress the contrast gracefully. As such, the infrared multi-scale retinex can be defined as Equation 34.

$$g(x,y) = F\left(\sum_{s=1}^{S} W_s \left\lfloor G\left(\frac{f(x,y)}{F_s(x,y) * f(x,y)}\right)\right\rfloor\right) \tag{35}$$

f(x,y) represents the input image while g(x,y) represents the output image. F and G are normalizing functions:

$$G(z(x,y)) = \rho \frac{z(x,y) - \min(z(x,y))}{\max(z(x,y)) - \min(z(x,y))} \tag{36}$$

$$F(z(x,y)) = L_{\max} \frac{z(x,y) - \min(z(x,y))}{\max(z(x,y)) - \min(z(x,y))} \tag{37}$$

ρ represents the maximum bit depth one wishes to have the ratio image to have. Through experimentation, acceptable values of ρ lie between 2^{10} and 2^{14}. By increasing ρ, one increases the number of bits of relevant edge information and as such, increases the chance of that edge appearing in the final image.

Chapter 5

METHODS

The quantitative analysis was performed in distinct stages. An overview of the process can be seen in Figure 8.

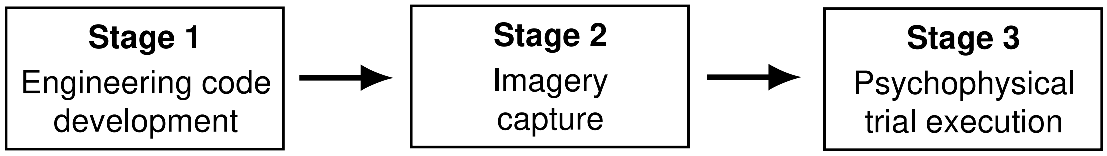


Figure 8 – Overview of the quantitative assessment method

The first stage was the development of engineering code representative of the four infrared contrast enhancement algorithms. The engineering code was developed in the MATLAB development environment. The purpose of this engineering code was to take digital frames that contain 16 bit pixels as input and output 8 bit imagery using an algorithm of choice for image enhancement. This input imagery took the form of single or multiple frames and output images or video, respectively. Additionally, this engineering code has the ability to run a rate of growth analysis on the code as described in Section 3.1. The analysis was run on an arbitrarily resized example infrared image. To facilitate use of the algorithms, a graphical user interface was developed for each algorithm. The code for each algorithm can be found in Appendix A.

The second stage involved the collection of digital frames captured from the infrared imager. To do so, a digital frame grabbing station was constructed. The station consisted of three components: an infrared imager (the BAE Systems SCC500H), capturing equipment (dedicated frame grabber and all necessary components), and a computer to control the capturing equipment. Using this setup, digital frames containing pixels output from the focal plane array on the camera were generated.

The imager was used in a variety of conditions that one can expect it to perform in. For this analysis, three scenes were selected: two artificial and one natural. It should be noted that the artificial scenes contain natural items but it was specifically designed to exhibit an arbitrary range of infrared behavior. To confirm the temperature range in the scene, an infrared thermometer was used to confirm the temperatures of objects with known emissivities. With known emissivities, the temperature recorded would be accurate.

The first scene involved a frame sequence where the imager captures a scene containing low-contrast objects (the thermal differential across the scene is less than one degree Fahrenheit) and then pans to a scene containing high-contrast objects (the thermal differential across the scene is greater than twenty degrees Fahrenheit). For this scene, a lit hibachi grill was used as the high contrast target while foliage, shortly after a rain event, provided the low-contrast scene elements. The second scene involved a frame sequence where the imager captures a scene containing low-contrast objects and then pans to a scene containing mid-contrast objects (the thermal differential across the scene is greater than one degree but less than twenty degrees Fahrenheit). For this scene, the hibachi grill was doused to provide the mid-contrast target while freshly rained on foliage provided the low-contrast target. Pictures of the representative targets can be seen in Figure 9.

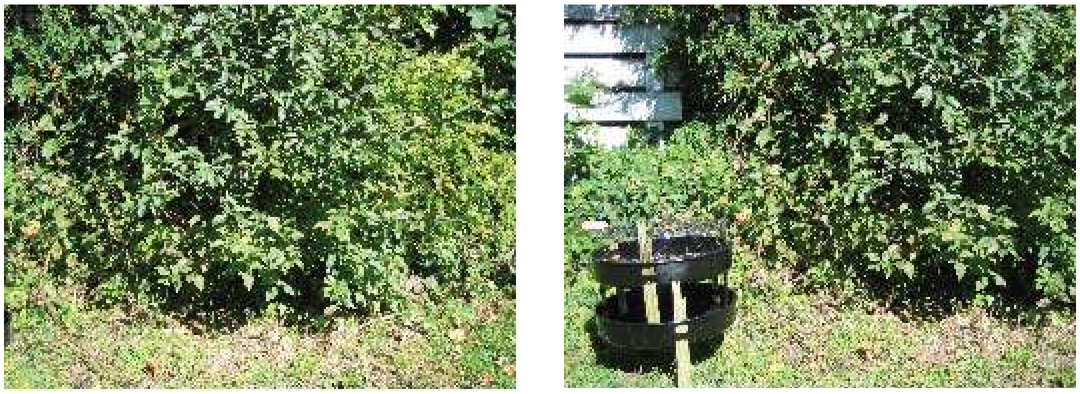


Figure 9 – Visible imagery of the backyard scene

The final scene involved a frame sequence where the imager captured a scene containing natural elements. It should be noted that the thermal differential across this scene was not noted. For this sequence, the imager was pointed at an interstate interchange with full view of objects such as vehicles, roads, lightposts, and foliage. Frames were recorded as vehicles passed the imager. An example visible picture from this scene can be seen in Figure 10.

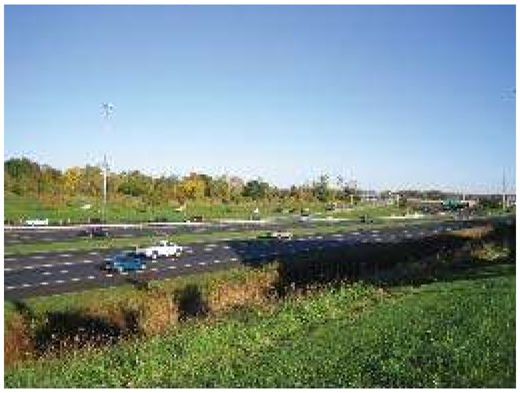


Figure 10 – Visible imagery of the highway scene

The third stage of the analysis involved the execution of paired-comparison psychophysical trials where test subjects were asked to evaluate processed video from the four different algorithms studied using the three scenes captured, resulting in 18 pairs to observe. For this research, the test subjects were separated into two groups: members of the Rochester City Fire Department and students from the Imaging Science program at the Rochester Institute of Technology. Since firefighters are one of the principal users of the infrared imager, their opinions are extremely important to the sponsor of this research. As a basis for comparison, the trial was also open to imaging science students. For each of the 18 pairs, observers were asked to answer two questions: "Of the two videos, select which one has the best quality" and "Of the two videos, select which one has the most detail." By asking the observer these specific questions, it was hoped that quantitative evidence would be generated to support the hypothesis that there is no difference between quality and detail between algorithms and computational considerations could be the deciding factor for the comparisons. To facilitate data collection and analysis, a MATLAB program was created to automate the process. Using this program, a calculation of the average ranks and interval scale values was generated according to the methods described in Section 3.2.

Chapter 6

RESULTS

After the research had been completed, the results were analyzed quantitatively. Using the means described in Sections 3 and 5, data collected from experimentation was analyzed to help determine the best infrared contrast enhancement algorithm. Once the results have been generated, a true ranking will be calculated for each quantitative comparison. This true ranking will account for any ranks that cannot be considered due to statistical limitations imposed on the results.

6.1 Subjective Image Analysis

After image collection, one of the first analyses that was carried out is a subjective analysis of the processed captured imagery. According to the previously described method, three scenes were collected: a low-to-high contrast scene, a low-to-mid contrast scene, and a natural scene. The low-to-high contrast scene was gathered by situating the infrared camera in a backyard that contained a fence, grass, trees, and flowers. The scene also contained a small hibachi grill that contained a small wood fire. As the video stream starts, the camera is facing a tree and some large bushes. As time progressed, the camera panned to the left, allowing the lit grill to enter the field of view. Finally, the camera panned right, returning to the original scene. Results of infrared contrast enhancement on frames containing the large tree and the lit grill can be seen in Figures 11 and 12.

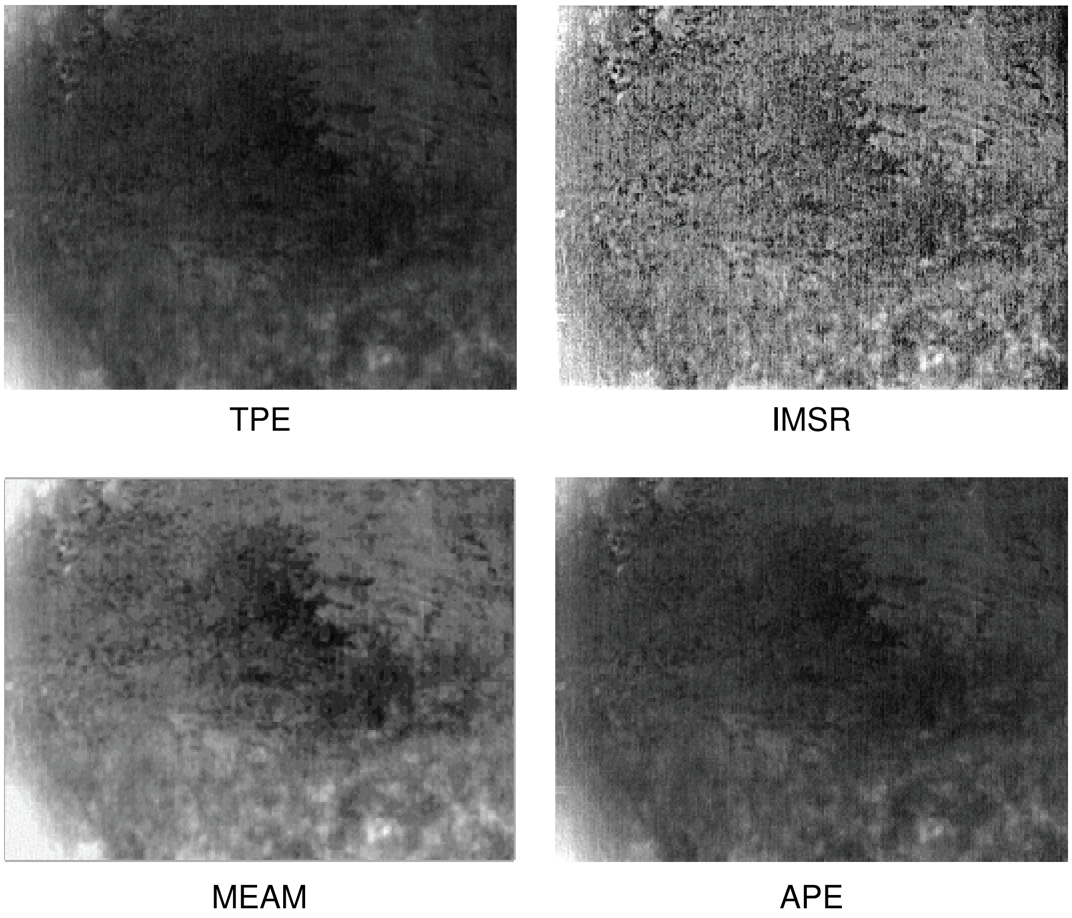


Figure 11 – Processed imagery of the low-to-high contrast scene containing foliage

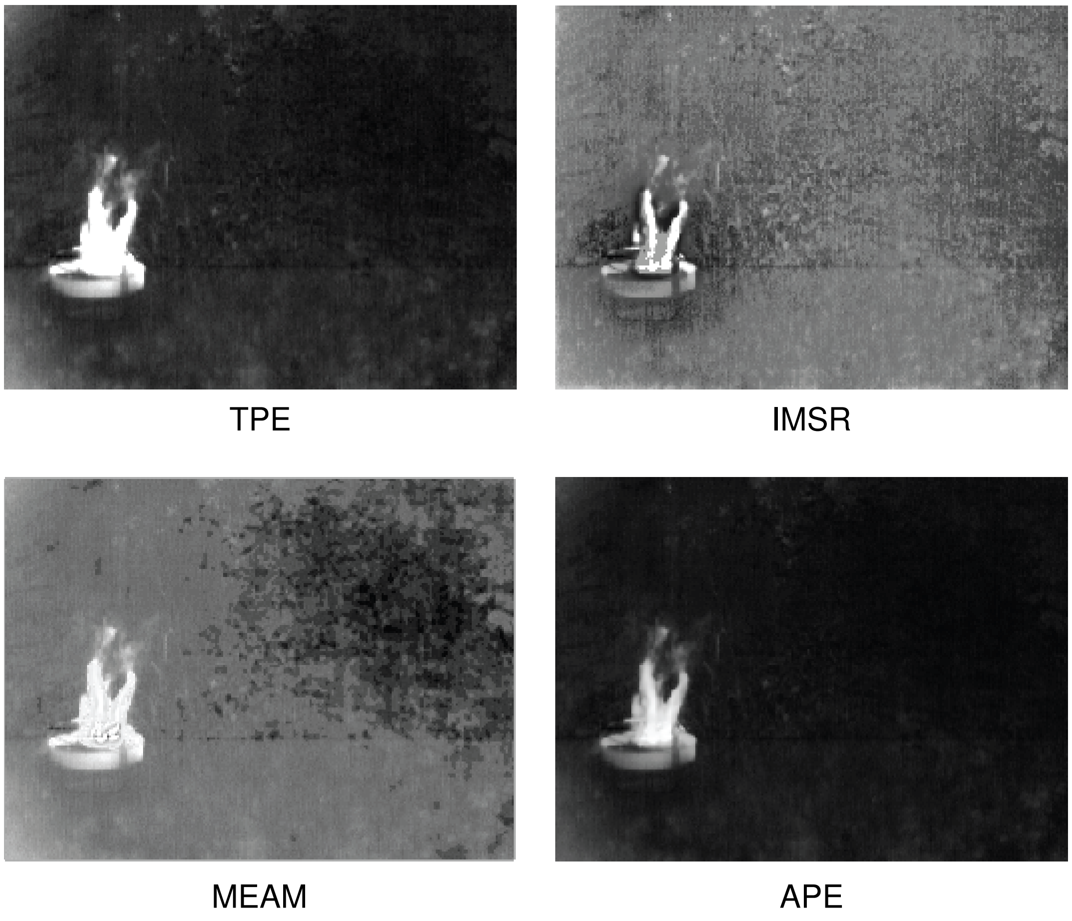


Figure 12 – Processed imagery of the low-to-high contrast scene containing the lit grill

As one can see from Figure 11, the low contrast trees and bushes do appear through each algorithm's processing yet each rendition shows noticeable differences. For example, IMSR has provided an image that has extremely high contrast but also has an appreciable amount of noise. MEAM has provided an image that has less contrast than IMSR but also less noise. APE continues this trend as there is less contrast than the two previous algorithms but also infinitesimal noise. Finally, TPE shows the least contrast amongst the scene but has no detectable noise. This informal ranking can be applied to the imagery when the scene changes, as in Figure 12. IMSR provides an image that shows structure in both the hot fire and the cool background. MEAM shows a bit less detail in the background while retaining a fair amount of detail in the fire. APE loses much of the detail in the background while retaining some in the flames of the fire itself. Finally, TPE loses the most detail in the background and a fair amount in the flames of the fire.

The low-to-mid contrast scene was very similar to the low-to-high contrast scene. Every component of the video stream was the same as the low-to-high contrast video with the exception that the hibachi grill was cooled after dousing the wood fire. Results of infrared contrast enhancement on frames containing the large tree and the unlit grill can be seen in Figures 13 and 14.

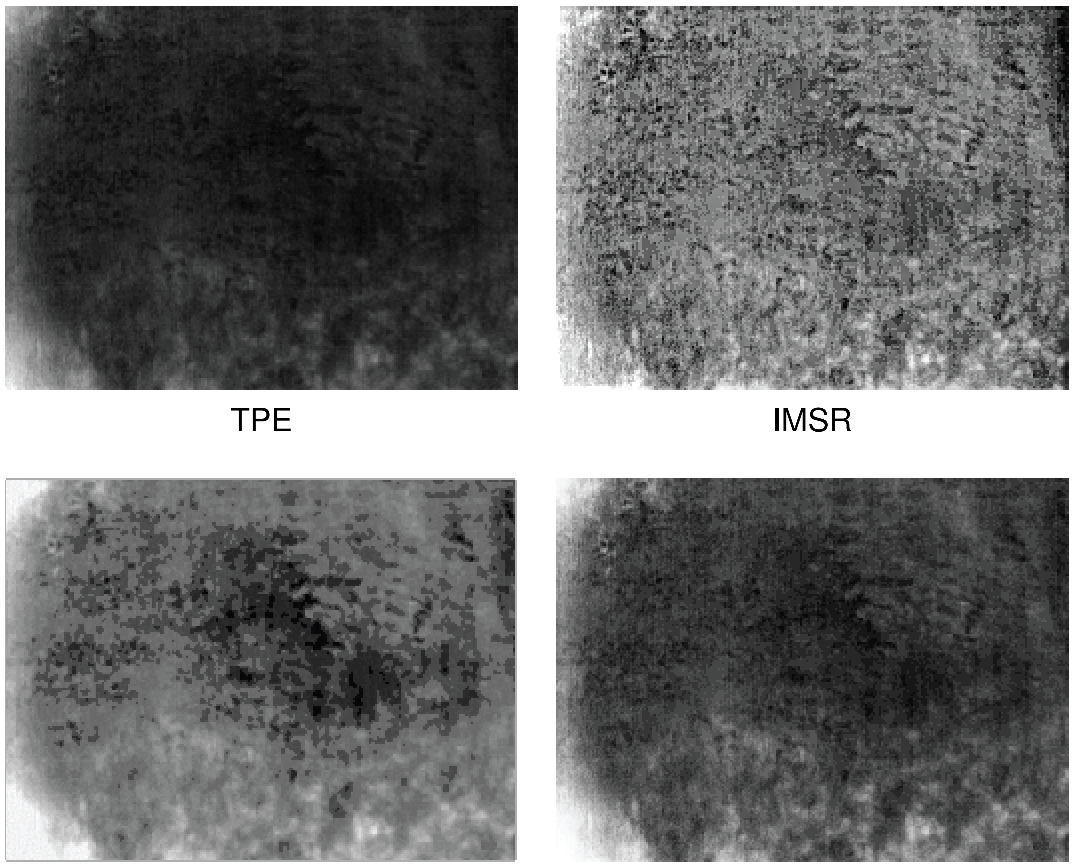


Figure 13 – Processed imagery of the low-to-mid contrast scene containing foliage

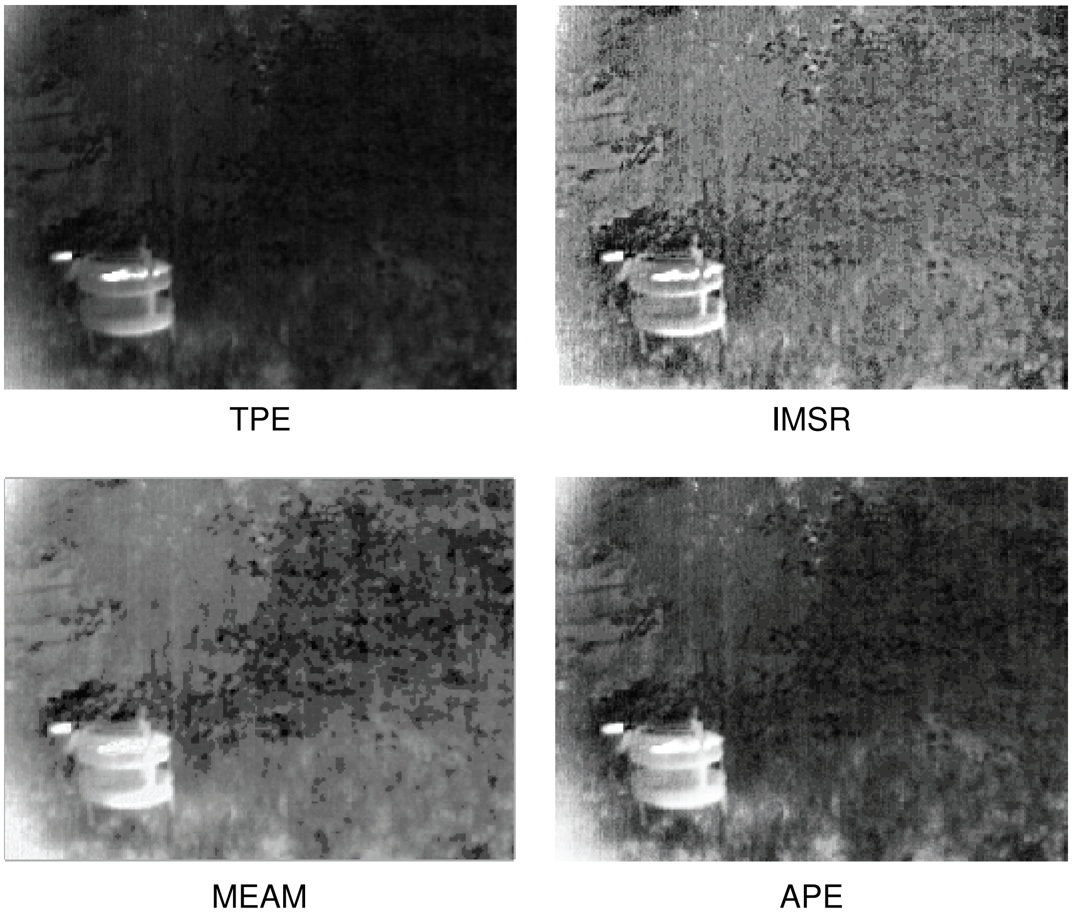


Figure 14 – Processed imagery of the low-to-mid contrast scene containing the unlit grill

As one can see in Figure 13, the same descriptions applied to Figure 11 can be applied with the beginning of this sequence of imagery as well. Some differences become apparent in Figure 14. IMSR provided an image that has the most overall contrast; structure can be seen in the grill, fence, and surrounding foliage. MEAM also has a significant amount of contrast overall but some detail was lost in the grass and tree. APE continues this trend as more detail was lost in the foliage. Finally, TPE loses much of the background contrast.

The natural scene consisted of situating the infrared camera in front of the intersection of East Henrietta Road and Interstate 390 in Brighton, a township located just outside downtown Rochester. While recording a video stream, the camera was kept stationary, allowing the motion of the vehicles on the roadways to provide changing infrared content. The results of image processing can be seen in Figure 15.

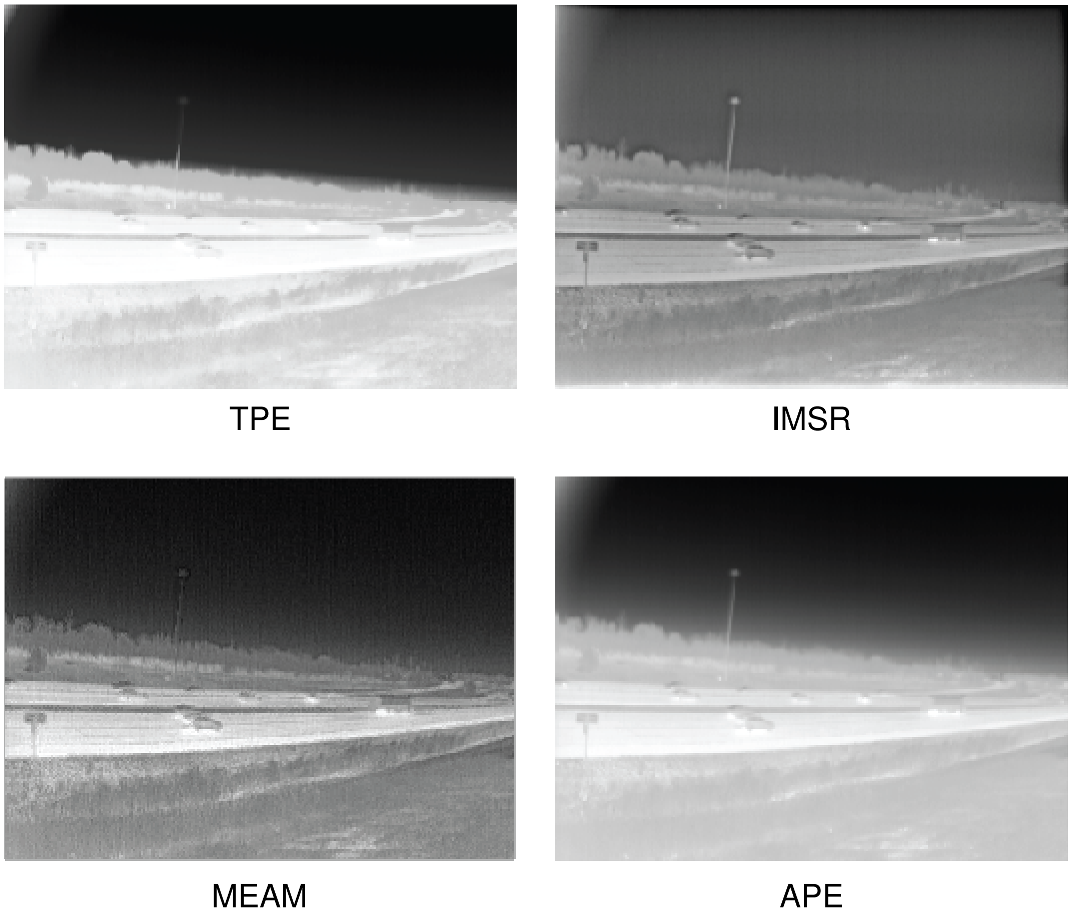


Figure 15 – Processed imagery of the natural scene

As one can see in Figure 15, the reconstructions of the scene vary widely across algorithms. TPE provided an image with stark contrast between the ground and the sky, losing most of the detail between the two fields. In addition, it becomes difficult to detect the tall light pole in the center of the scene. The other three algorithms do not exhibit this behavior. IMSR provides a scene where much detail can be seen in the ground plane at the expense of some of the detail in the sky. MEAM provided a more balanced rendition of the scene while APE provided an image that was slightly better in rendition than TPE.

6.2 Algorithm Analysis

As described in Section 5, a rate of growth analysis was conducted on each algorithm. This entailed recording the average execution time for each algorithm, repeated an arbitrary number of times, versus the size of the input image to each algorithm. It should be noted that each algorithm was executed on a computer system with swap memory enabled. By measuring the execution time on this system, there is a chance that swap access time may be included in the measurement of execution time if the system did not have enough physical memory to complete the algorithm. Due to no disk accesses being noticed by the principal investigator during the efficiency test, it is assumed that the swap was not accessed. Hence, the execution times recorded are for algorithmic execution alone. The results of this analysis for each algorithm can be seen in Figure 16.

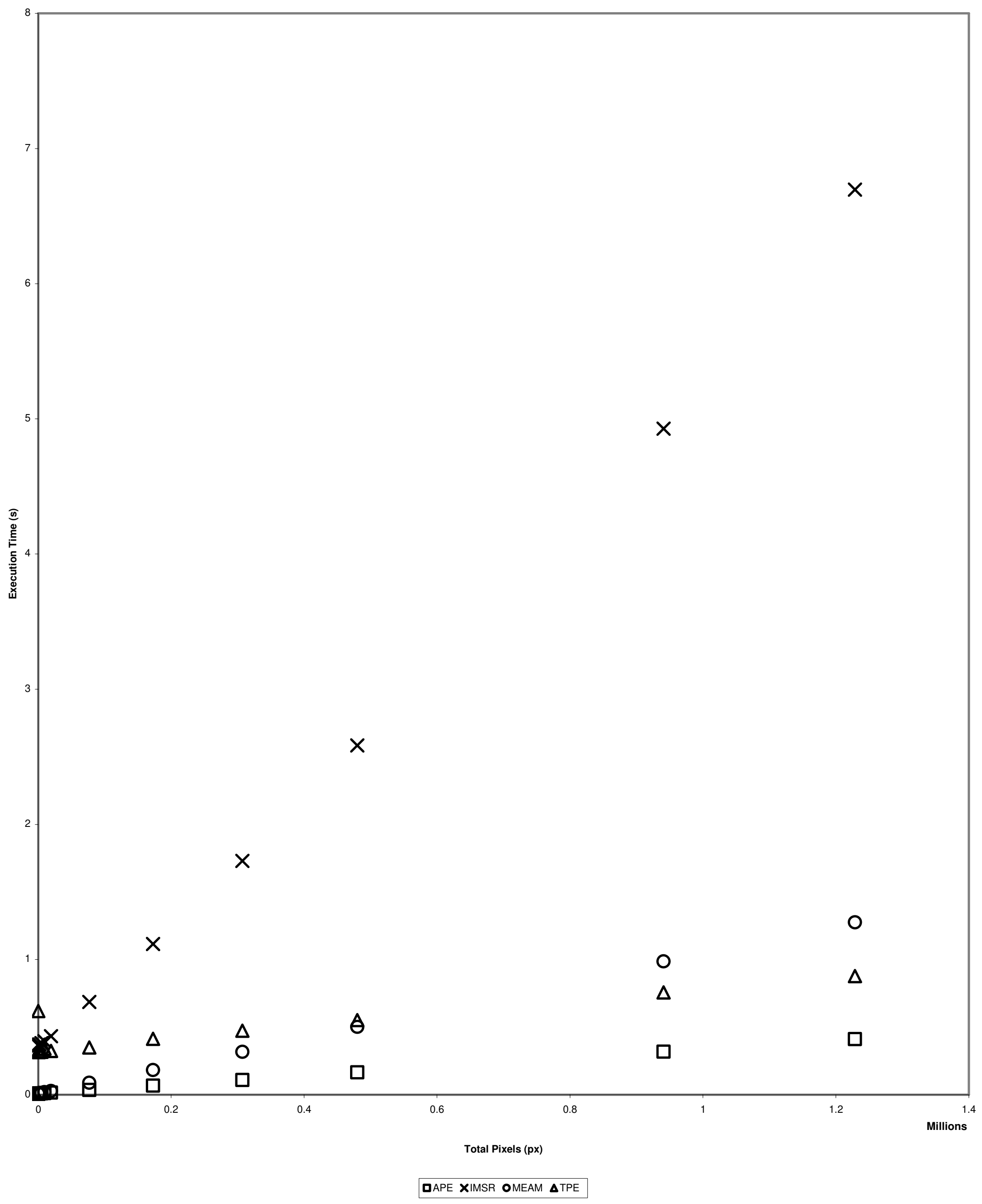


Figure 16 – Rate of growth analysis for all algorithms

As one can see, APE, TPE, and MEAM all have similar operating characteristics that cannot be easily distinguished. That can be explained by the linear nature of each algorithm's components. However, IMSR takes on a character that could not be accurately assessed when viewed alone: the rate of growth has a curved nature to it, hence it is not linear. That is fairly easy to understand as the Fourier transform is a component of IMSR. As image size increases, the computational load of the Fourier transform increases logarithmically, not linearly. As such, IMSR will have a higher rate of growth since it includes both a logarithmic and linear nature. Hence, if one were to rank the results based on these results, one would find the ranking found in Table 3.

Algorithm	APE	IMSR	MEAM	TPE
Rank	1	2	1	1

Table 3 – Rank based on rate of growth analysis

6.3 Paired Comparison Testing

As described in Section 5, a group of firefighters and students were sought to participate in a paired comparison psychophysical trial utilizing the three collected scenes. Members of the Rochester Fire Department and Imaging Science department participated in the trials over the course of three months. Groups were formed based on the scheduling in place and as such, data could not be collected at the same time. The principal investigator presented two scenes to the firemen during one week and presented the final scene during another week. As a result, the paired comparison trial of the first two scenes involved thirteen observers while the final scene involved sixteen observers. For the students, all scenes were presented at the same time to ten observers.

Using developed software, observers were presented with exemplar video from each of the four algorithms and were asked to select video that exhibited individually the best quality and most detail. Through background calculations, an average rank and interval scale were developed according to the methods described in Section 3.

6.3.1 Average Rank

Using the data collected from the paired comparison experiment, one can create a quick quantitative metric called an average rank. An average rank was calculated for three distinct groups: the firefighters, the imaging scientists, and a combined group. To calculate an average rank, one uses the process described in Section 3 and Equation 2. Once calculated, the algorithm that has the highest average rank is the best. From Section 5, one can see that six average ranks are calculated: a rank for each question asked of each scene. A plot of average rank versus algorithm can be seen in Figures 17 through 22.

As there exists the possibility that one cannot discern the average rank visually, one can infer that there is no statistical difference between the rank of one algorithm compared to another. This lack of difference is important because without a difference, it would be impossible to determine the true ranking; all that could be done is assume the same rank for those algorithms not deemed statistically different. To confirm statistical difference, the t test, as described in Section 3, was employed. For each question in each scene, a test hypothesis was formed for each possible pair that could be formed from the statistics generated for each algorithm. As such, a test statistic and appropriate comparison T curve value were found for each pair. The results can be seen in Tables 4 through 9. The numerical values in each table for the pairs indicated match a specific algorithm (1 – APE, 2 – IMSR, 3 – MEAM, 4 – TPE). It should be noted that the t-test was only performed on the combined data set because the final determination of best algorithm was made only on this set. As such, split results for both groups are presented separately for completeness.

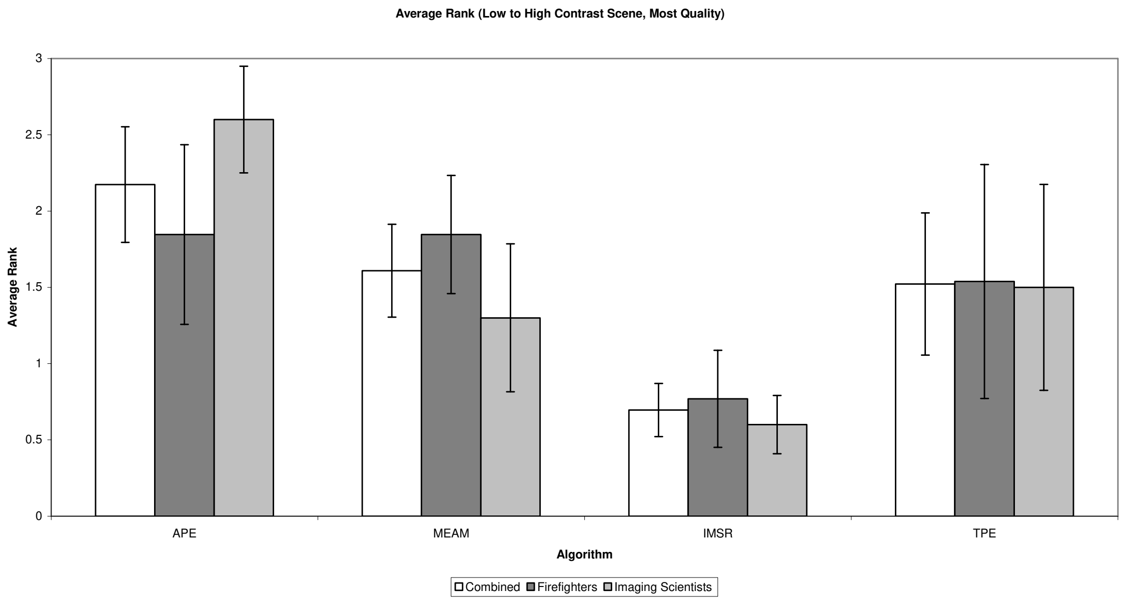


Figure 17 – Average rank using the low-to-high contrast scene concerning best quality

Pairs	1 & 2	1 & 3	1 & 4	2 & 3	2 & 4	3 & 4
t	2.155806	6.264736	2.236068	4.162331	0.312348	3.254126
df	43	38	43	40	42	36
T value	2.016692	2.024394	2.016692	2.021075	2.018082	2.028094
Reject H_0	Y	Y	Y	Y	N	Y

Table 4 – T test results for the low-to-high contrast scene involving best quality

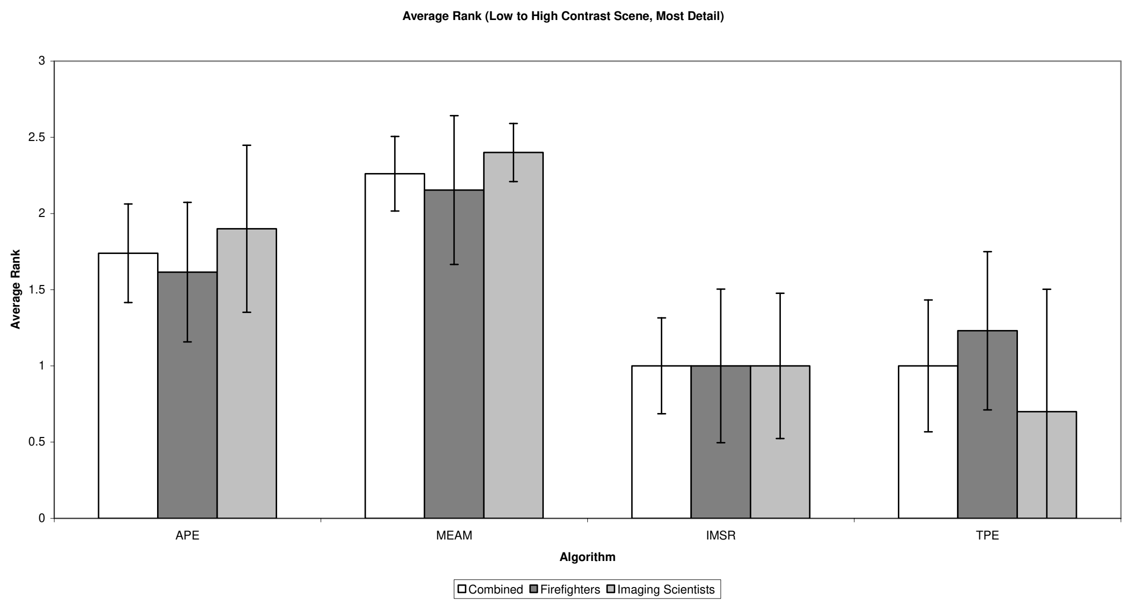


Figure 18 – Average rank using the low-to-high contrast scene concerning most detail

Pairs	1 & 2	1 & 3	1 & 4	2 & 3	2 & 4	3 & 4
t	2.184281	2.919381	2.681848	5.318883	4.833333	0
df	43	43	43	43	40	42
T value	2.016692	2.016692	2.016692	2.016692	2.021075	2.018082
Reject H_0	Y	Y	Y	Y	Y	N

Table 5 – T test results for the low-to-high contrast scene involving most detail

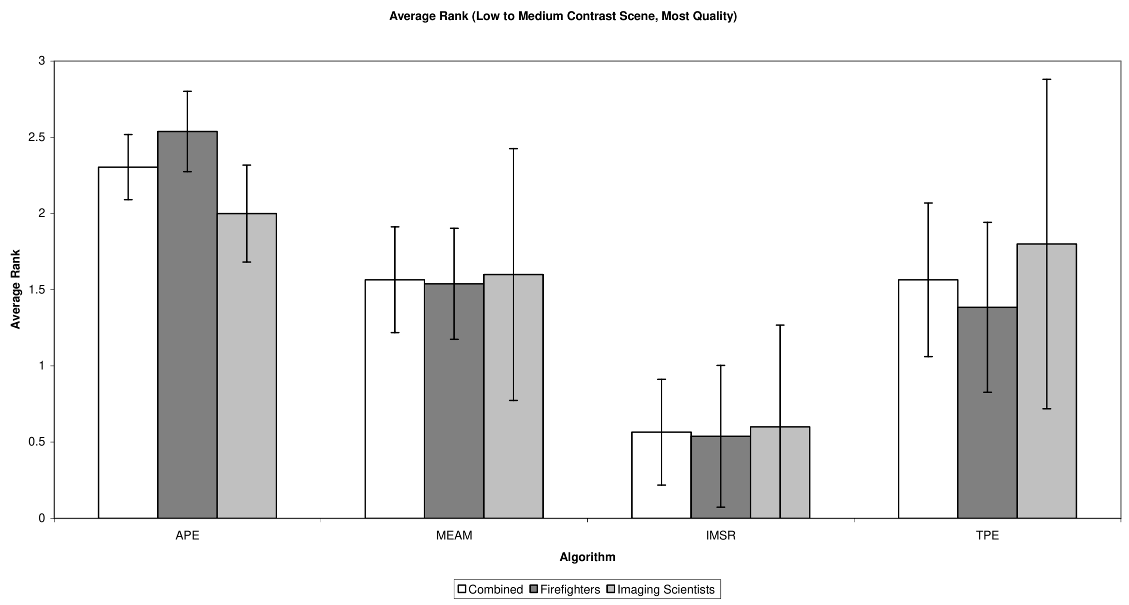


Figure 19 – Average rank using the low-to-mid contrast scene concerning best quality

Pairs	1 & 2	1 & 3	1 & 4	2 & 3	2 & 4	3 & 4
T	3.113209	7.325199	2.75119	3.78583	0	3.418295
df	41	41	37	43	42	42
T value	2.019541	2.019541	2.026192	2.016692	2.018082	2.018082
Reject H_0	Y	Y	Y	Y	N	Y

Table 6 – T test results for the low-to-mid contrast scene involving best quality

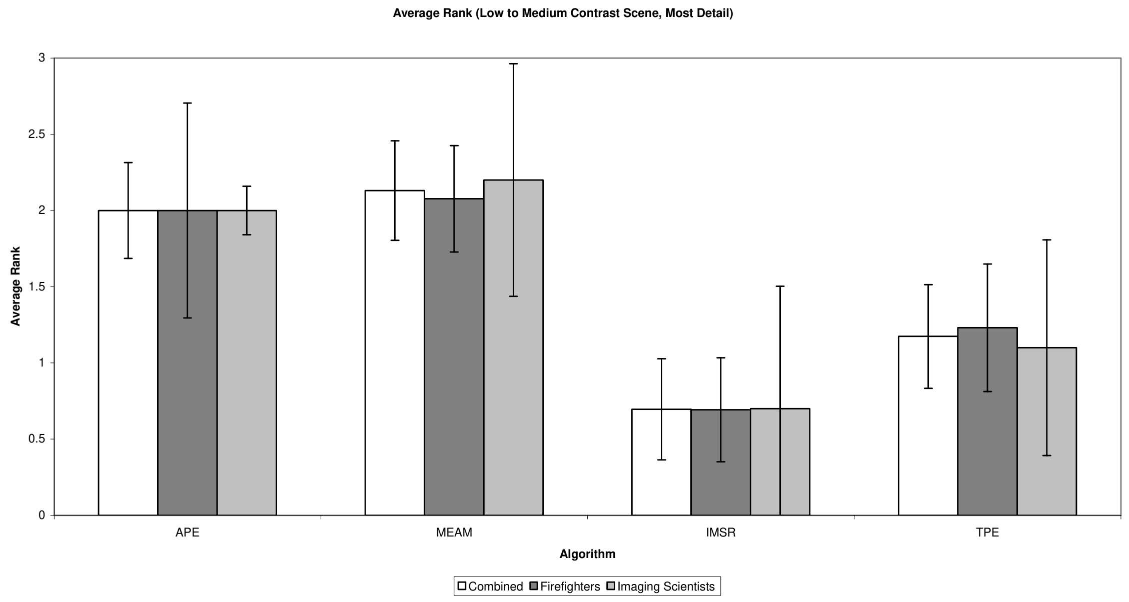


Figure 20 – Average rank using the low-to-mid contrast scene concerning most detail

Pairs	1 & 2	1 & 3	1 & 4	2 & 3	2 & 4	3 & 4
t	0.513809	5.117663	3.219961	5.578018	3.694764	1.840317
df	43	43	43	43	43	43
T value	2.016692	2.016692	2.016692	2.016692	2.016692	2.016692
Reject H_0	N	Y	Y	Y	Y	N

Table 7 – T test results for the low-to-mid contrast scene involving most detail

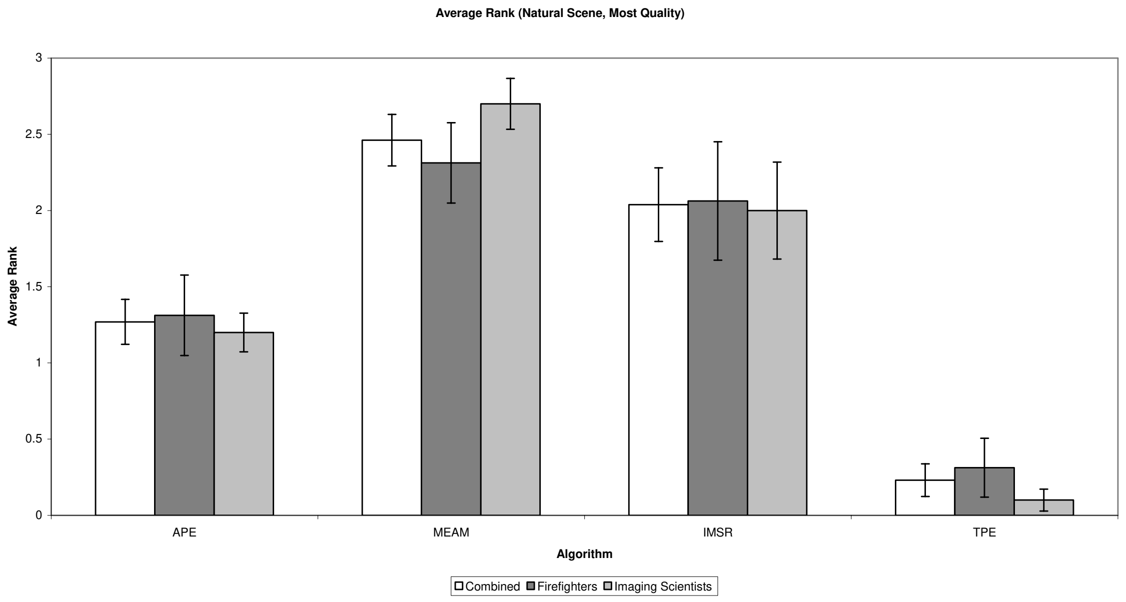


Figure 21 – Average rank using the natural scene concerning best quality

Pairs	1 & 2	1 & 3	1 & 4	2 & 3	2 & 4	3 & 4
t	6.870254	3.996804	6.67532	2.139252	13.76279	9.921708
Df	49	47	48	48	47	43
T value	2.009575	2.011741	2.010635	2.010635	2.011741	2.016692
Reject H_0	Y	Y	Y	Y	Y	Y

Table 8 – T test results for the natural scene involving best quality

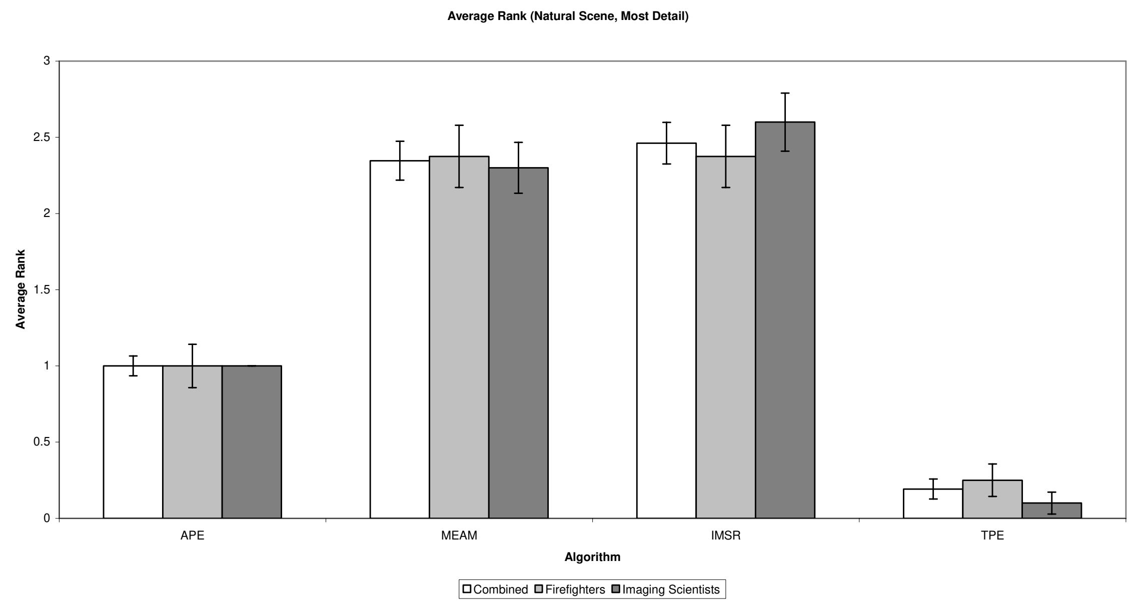


Figure 22 – Average rank using the natural scene concerning most detail

Pairs	1 & 2	1 & 3	1 & 4	2 & 3	2 & 4	3 & 4
t	9.955402	10.55556	7.263002	0.727607	15.90293	16.36366
df	45	44	49	49	45	44
T value	2.014103	2.015368	2.009575	2.009575	2.014103	2.015368
Reject H_0	Y	Y	Y	N	Y	Y

Table 9 – T test results for the natural scene involving most detail

The first scene to be analyzed was the low-to-high contrast scene. As one can see in Figure 17, the average rank from best to worst when considering quality was APE, MEAM, TPE, and IMSR. From the plot, MEAM and TPE exhibit similar rankings from visual judgment; this is confirmed from Table 4. The test statistic value for the comparisons between MEAM and TPE were not great enough to reject the null hypothesis. As such, one must consider that the two algorithms have the same rank. Based on these results, the exhibited rank in answering this question can be seen in Table 10.

Algorithm	APE	IMSR	MEAM	TPE
Rank	1	3	2	2

Table 10 – True rank based on rank analysis for quality in the low-to-high contrast scene

As one can see in Figure 18, the average rank for detail differed from the average rank for quality. From best to worst, the ranking is MEAM, APE, TPE, and IMSR. However, from Table 5, one can see that there is no statistical difference between IMSR and TPE. Due to this, the rank must be considered the same for each algorithm. Based on these results, the exhibited rank in answering this question can be seen in Table 11.

Algorithm	APE	IMSR	MEAM	TPE
Rank	2	3	1	3

Table 11 – True rank based on rank analysis for detail in the low-to-high contrast scene

The second scene to be analyzed was the low-to-mid contrast scene. As one can see in Figure 19, the average rank from best to worst when considering quality was APE, MEAM, TPE, and IMSR. From the plot, MEAM and TPE exhibit similar rankings from visual judgment; this is confirmed from Table 6. The test statistic value for the comparison between TPE and MEAM was not great enough to reject the null hypothesis. As such, one must consider that these two algorithms have the same rank. Based on these results, the exhibited rank in answering this question can be seen in Table 12.

Algorithm	APE	IMSR	MEAM	TPE
Rank	1	3	2	2

Table 12 – True rank based on rank analysis for quality in the low-to-mid contrast scene

As one can see in Figure 20, the average rank when concerned with detail differed from the average rank when concerned with quality. From best to worst, the ranking is

MEAM, APE, TPE, and IMSR. However, from Table 7, one can see that there is no statistical difference between APE & MEAM and between IMSR & TPE. Due to this, the rank must be same for each. Based on these results, the exhibited rank in answering this question can be seen in Table 13.

Algorithm	APE	IMSR	MEAM	TPE
Rank	1	2	1	2

Table 13 – True rank based on rank analysis for detail in the low-to-mid contrast scene

The final scene to be analyzed was the natural scene. As one can see in Figure 21, the average rank from best to worst when considering quality was TPE, APE, MEAM, and IMSR. From the plot, one can see that there is definitive visual evidence that there is separability between the average ranks of the algorithms. The values in Table 8 confirm this as well. Based on these results, the exhibited rank in answering this question can be seen in Table 14.

Algorithm	APE	IMSR	MEAM	TPE
Rank	3	2	1	4

Table 14 – True rank based on rank analysis for quality in the natural scene

As one can see in Figure 22, the average rank when concerned about detail is more ambiguous when compared to the ranks for quality. From best to worst, the ranking is IMSR, MEAM, APE, and TPE. From the plot, it is visually apparent that IMSR and MEAM share the same rank. Table 9 confirms this finding. As such, the rank must be same for IMSR and MEAM. Based on these results, the exhibited rank in answering this question can be seen in Table 15.

Algorithm	APE	IMSR	MEAM	TPE
Rank	2	1	1	3

Table 15 – True rank based on rank analysis for detail in the natural scene

6.3.2 Interval Scale

In addition to the average rank, a quantitative metric called an interval scale can be calculated from the psychophysical data. Using the procedures described in the Sections 3 and 5, an interval scale was calculated for each question asked of the three scenes, resulting in six scales. Error metrics were calculated using Equation 9. A plot of interval scale versus algorithm for each respective scene and question can be seen in Figures 23 through 28 with their respective error bars.

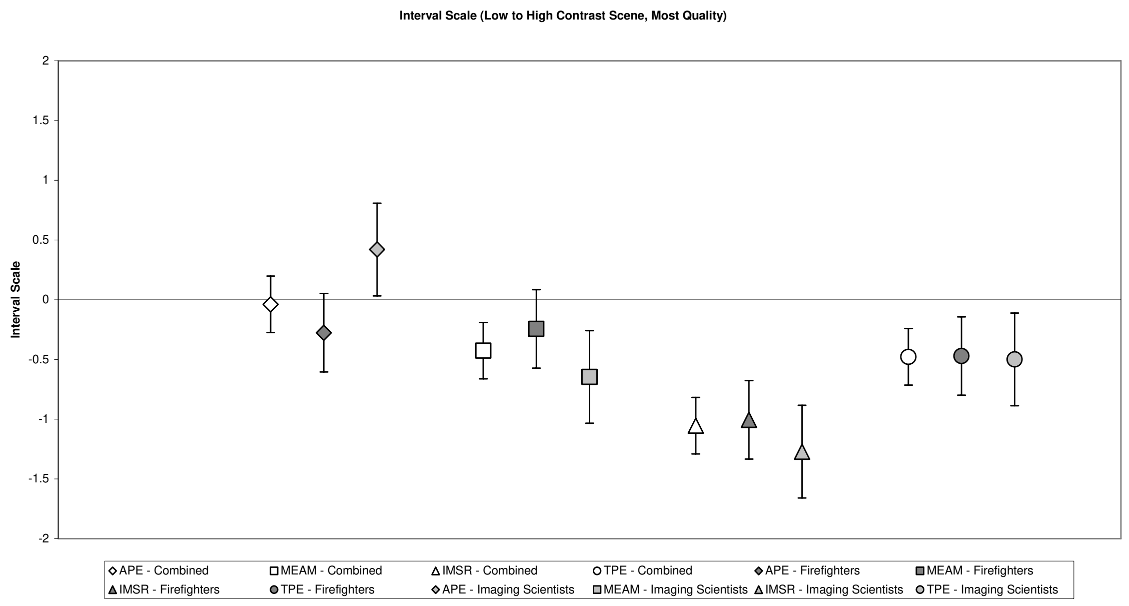


Figure 23 – Interval scale for the low-to-high contrast scene involving best quality

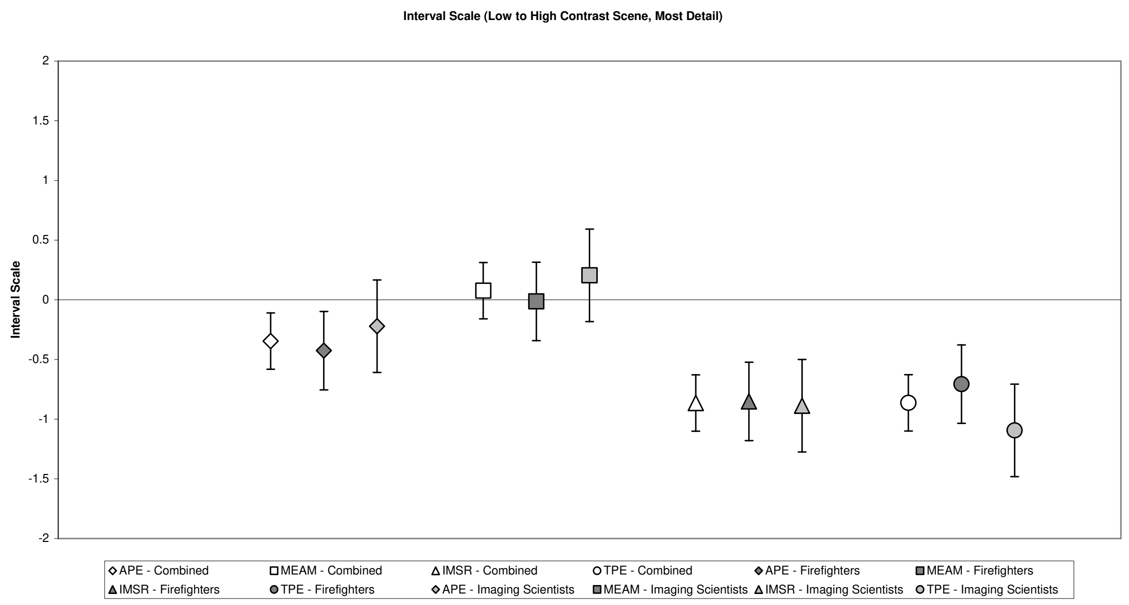


Figure 24 – Interval scale for the low-to-high contrast scene involving most detail

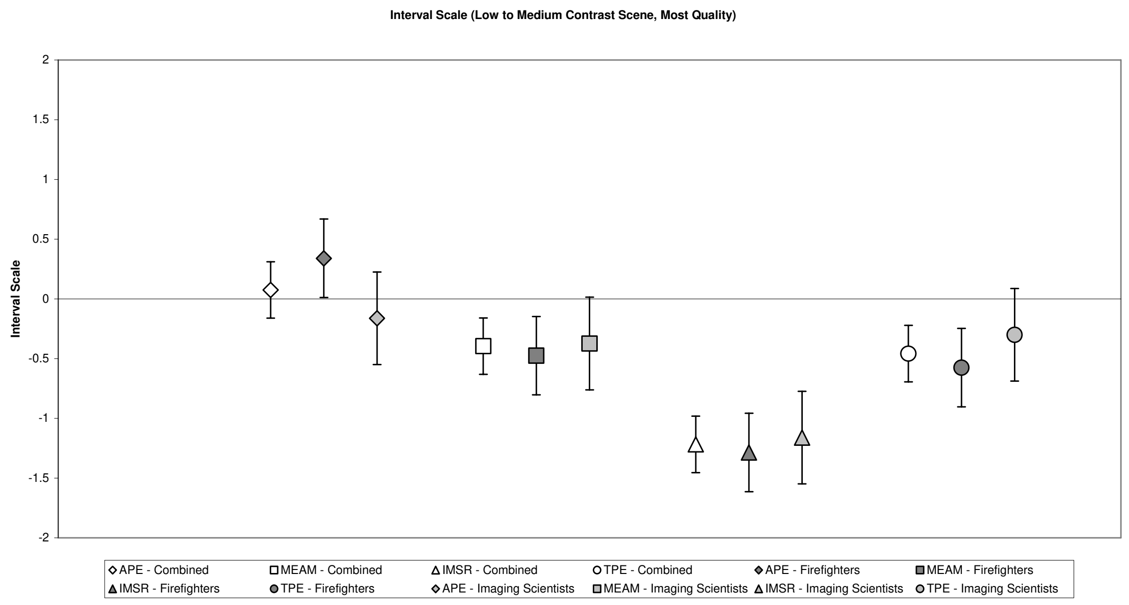


Figure 25 – Interval scale for the low-to-mid contrast scene involving best quality

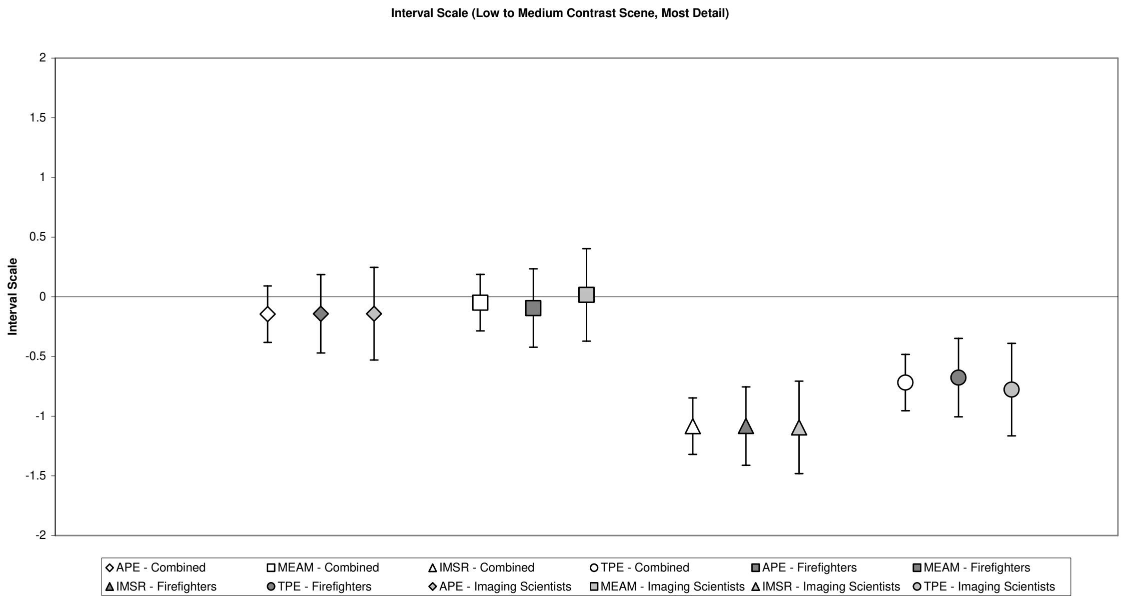


Figure 26 – Interval scale for the low-to-mid contrast scene involving most detail

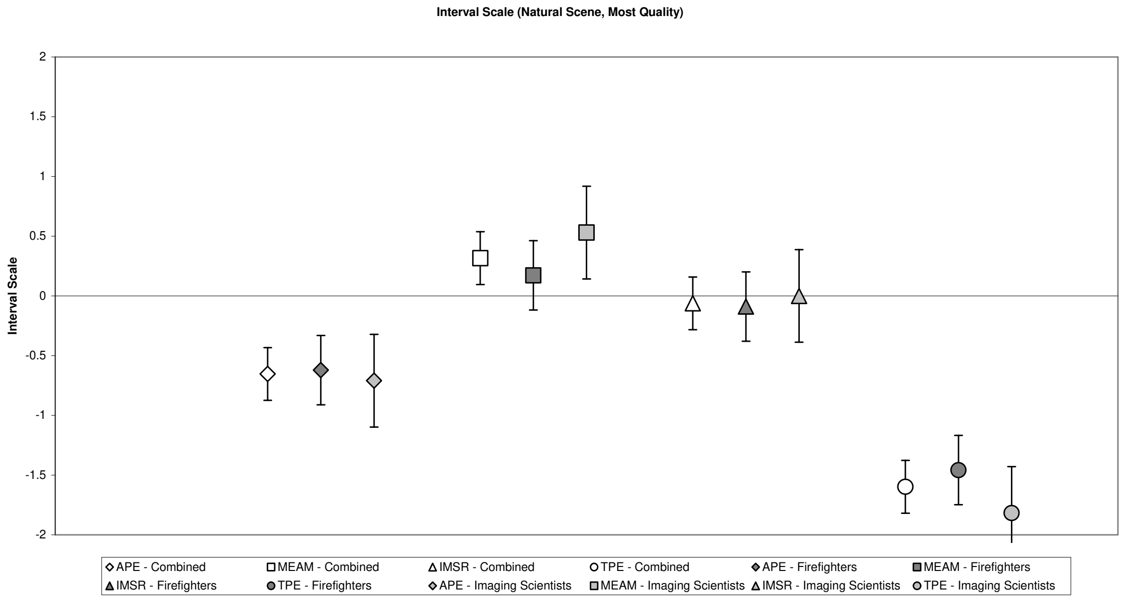


Figure 27 – Interval scale for the natural scene involving best quality

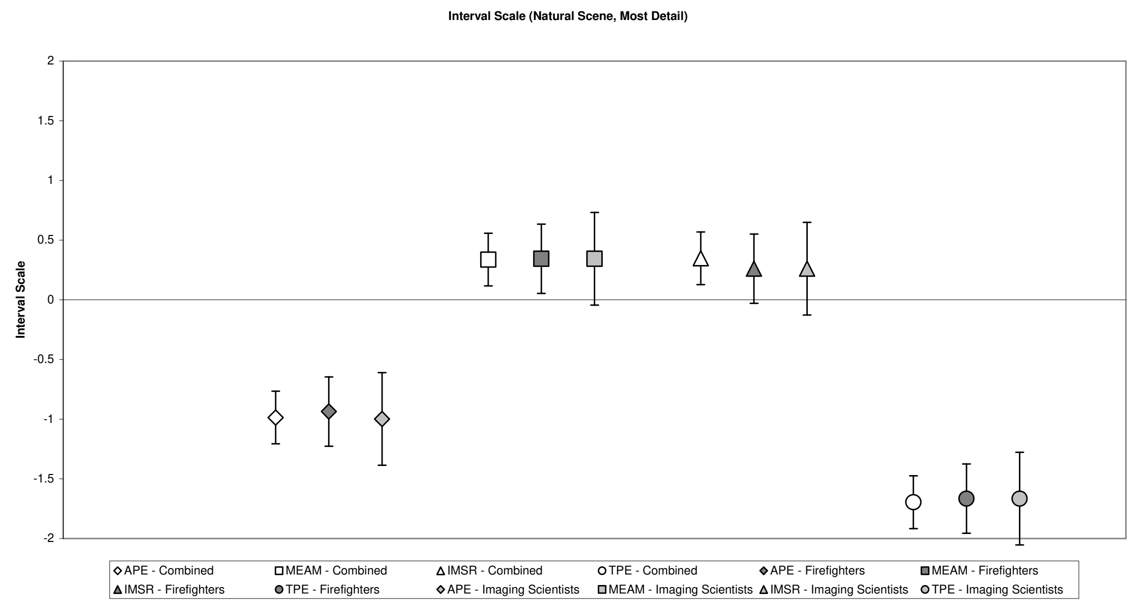


Figure 28 – Interval scale for the natural scene involving most detail

The first scene to be analyzed was the low-to-high contrast scene. Figure 23 illustrates that the interval scale ranks APE, MEAM, TPE, and IMSR from best to worst when concerned with quality. Also, one can see that the interval scale for TPE lies in the error bars of MEAM. As such, the interval scale values for MEAM and TPE are not statistically different and as such, must be considered the same. If so, the true rank would be seen as Table 16.

Algorithm	APE	IMSR	MEAM	TPE
Rank	1	3	2	2

Table 16 – True rank based on scale analysis for quality in the low-to-high contrast scene

Figure 24 shows that the interval scale ranks MEAM, APE, IMSR, and TPE from best to worst when concerned with detail. However, as seen in the figure, there is significant overlap with the error bars of the interval scale for IMSR and TPE. As such, their respective rankings must be considered the same. As these two algorithms are separable from the other two, the final ranking for this question can be found in Table 17.

Algorithm	APE	IMSR	MEAM	TPE
Rank	2	3	1	3

Table 17 – True rank based on scale analysis for detail in the low-to-high contrast scene

The second scene to be analyzed was the low-to-mid contrast scene. From Figure 25, one can see that the ranking from best to worst is AP, MEAM, TPE, and IMSR when concerned with quality. One should also see that there is significant overlap between the error bars of MEAM and TPE. As such, the rankings for these two algorithms must be considered the same. Therefore, the true ranking for this question can be seen in Table 18.

Algorithm	APE	IMSR	MEAM	TPE
Rank	1	3	2	2

Table 18 – True rank based on scale analysis for quality in the low-to-mid contrast scene

When concerned with detail, the results follow a different pattern. As one can see in Figure 26, the ranking from best to worst is MEAM, APE, TPE, and IMSR. One should also see that the scale values for APE and MEAM fall within their respective error bars. Additionally, the scale values for IMSR and TPE falls within their respective error bars. As such, the rank for the combination of APE and MEAM and the combination of IMSR and TPE are the same. Hence, the true rank using this question can be found in Table 19.

Algorithm	APE	IMSR	MEAM	TPE
Rank	1	3	1	2

Table 19 – True rank based on scale analysis for detail in the low-to-mid contrast scene

The final scene to be analyzed was the natural scene. From Figure 27, one can see that the best to worst ranking when concerned with quality is MEAM, IMSR, APE, and TPE. From the figure, one can also see that there is definite separation between the scale values. As such, the true rank for this question can be seen in Table 20.

Algorithm	APE	IMSR	MEAM	TPE
Rank	3	2	1	4

Table 20 – True rank based on scale analysis for quality in the natural scene

From Figure 28, one can see that this trend becomes slightly ambiguous. The best to worst ranking when concerned with detail is IMSR, MEAM, APE, and TPE. One should note that the scale value for MEAM falls within the error bars for IMSR. As such, their true ranking must be considered the same. Therefore, the true rank using the interval scale data can be seen in Table 21 for this question.

Algorithm	APE	IMSR	MEAM	TPE
Rank	2	1	1	3

Table 21 – True rank based on scale analysis for detail in the natural scene

Chapter 7

CONCLUSIONS

Dynamic range compression and contrast enhancement are two image processing methods that are highly important to any person designing an infrared imaging system. Often, the detector in an infrared system has a high dynamic range while the output display device has a much lower dynamic range. As such, an intermediate step must be taken to make these two components of the system compatible in such a way that is pleasing to the human observer.

This thesis strived to find examples of this intermediary step and quantitatively determine the feasibility and utility of each. By performing a rate of growth analysis on each algorithm, it became possible to compare the resources required by each algorithm in a system-independent fashion. By performing a psychophysical trial, it became possible to use the end user of an infrared system as an objective quantitative metric. Through careful analysis, it became possible to form a decision on which algorithm is the best to use in an infrared system.

The first step in analyzing the results was determining whether there was a difference between quality and detail. This is important because if there is a difference, a unique condition is placed on a system engineer designing the camera, namely the decision of which algorithm to use as the intermediary step becomes complex. Whereas before, if an algorithm was determined to have a high quality, the systems engineer would choose such an algorithm. However, if there is a difference between quality and detail, the system engineer must choose an algorithm that is appropriate to the application. For example, if the end user of an infrared system was a firefighter, their main concern would be whether they could distinguish between human beings or man-made objects. As such, spatial detail would be of primary importance since detail is what differentiates an ambiguous blob on the output display from a human being. If the end user of a system is a foot soldier, they would be more concerned with a noiseless output display. This is important because if a soldier sees a quick appearance of random pixels, he may perceive that to be an enemy and take inappropriate action. As such, a noise-free display would be a quality issue. From the results of the psychophysical experiment, the separation between quality and detail is apparent. For the most part, the true rankings for the question of detail were different than the true rankings for the question of quality, leading one to believe that quality and detail can be separate.

From the true rankings, one can also see that the best algorithm seems to be a frequency-based method: MEAM. MEAM was a clear winner when used in a natural, everyday scene with a high signal-to-noise ratio. However, more importantly, observers preferred MEAM just as favorably as APE when used in a low signal-to-noise ratio scene such as the artificial scenes. To determine which algorithm was the best, a true average rank was calculated using Tables 3 and 10 through 21. The results of this average can be seen in Table 22.

Algorithm			**APE**	**IMSR**	**MEAM**	**TPE**
Rate of Growth			1	2	1	1
Average Rank	Low-to-High	Quality	1	3	2	2
		Detail	2	3	1	3
	Low-to-Mid	Quality	1	3	2	2
		Detail	1	2	1	2
	Natural	Quality	3	2	1	4
		Detail	2	1	1	3
Interval Scale	Low-to-High	Quality	1	3	2	2
		Detail	2	3	1	3
	Low-to-Mid	Quality	1	3	2	2
		Detail	1	3	1	2
	Natural	Quality	3	2	1	4
		Detail	2	1	1	3
Total Rank			21	31	17	33
Average Rank			1.61	2.38	1.30	2.53
Final Rank			2	3	1	4

Table 22 – Final results

As one can see, the collated results show that from best to worst, the algorithm of choice is MEAM, APE, IMSR, and TPE. One of the goals of this research was to develop algorithms that were better than the baseline. From this study, that goal has been accomplished. With additional optimization for the hardware it is intended for, the frequency-based methods MEAM and IMSR should prove to be a superior algorithm for infrared contrast enhancement.

Chapter 8

FUTURE RESEARCH

After completing this thesis, the principal investigator found two areas of research that could be tended to in the future. The first area is a deep exploration of how each input parameter affects the performance of each infrared contrast enhancement algorithm. Due to the nature of the experiment, a single “one size fits all” parameter set was chosen for each algorithm to apply to each scene. In actual usage, it might be more beneficial to have a parameter or parameters that could be changed by an end user to enable the best display of infrared imagery. For example, by applying independent α and β parameters to each of the Gaussian fields in the IMSR algorithm, a smoother image might result.

The second area that future research can be performed in is in the exploration of how small changes to the current algorithms might be beneficial to the algorithm as a whole. For example, the principal investigator wanted to see if an adaptive attenuation of the highpass information in the MEAM algorithm would lead to a better overall quality. In theory, by adaptively attenuating the highpass information, a greater control over the contrast among low and high temperature edges can be achieved.

REFERENCES

[1] Blohm, W. Video dynamic range compression of portrait images by simulated diffuse scene illumination. Optical Engineering (35), January 1996, p255-261.

[2] Devore, J. L. Probability and statistics for engineering and the sciences. 5th Ed. Pacific Grove: Duxbury, 2000.

[3] Engeldrum, P. G. Psychometric scaling: a toolkit for imaging systems development. Winchester: Imcotek Press, 2000.

[4] Enkvist, M. and L. Haglund. Automatic dynamic range adaptation for image data. Proceedings of the SPIE: Visual Information Processing XII (5108), 2003, p171-180.

[5] Gonzalez, R. C. and R. E. Woods. Digital image processing. 2nd Ed. Upper Saddle River: Prentice Hall, 2002.

[6] Gruben, J. H., et al. Scene-based algorithm for improved FLIR performance. Proceedings of the SPIE: Infrared Imaging Systems: Design, Analysis, Modeling, and Testing XI (4030), 2000, p184-195.

[7] Jin, Y., L. Fayad, and A. Laine. Contrast enhancement by multi-scale adaptive histogram equalization. Proceedings of the SPIE: Wavelets: Application in Signal and Image Processing IX (4478), 2001, p206-213.

[8] Larson, G. W., H. Rushmeier, and C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. IEEE Transactions on Visualization and Computer Graphics (3), October 1997, p291-306.

[9] McConnell, J. J. Analysis of algorithms: an active learning approach. Boston: Jones and Bartlett Publishers, 2001.

[10] Montag, E. D. Empirical formula for creating error bars for the method of paired comparison. Journal of Electronic Imaging (15), January 2006, p105021-3/

[11] Montag, E. D. and M. D. Fairchild. Psychophysical evaluation of gamut mapping techniques using simple rendered images and artificial gamut boundaries. IEEE Transactions on Image Processing (6), July 1997, p977-89.

[12] Ngo, H., L. Tao, and V. Asari. Design of an efficient architecture for real-time image enhancement based on a luma-dependent nonlinear approach. Proceedings of the International Conference on Information Technology: Coding and Computing (1), 2004, p656-60.

[13] Rahman, Z., D.J. Jobson and G.A. Woodell. Retinex processing for automatic image enhancement. Journal of Electronic Imaging (13), January 2004, p100-110.

[14] Reinhard, E. and K. Devlin. Dynamic range reduction inspired by photoreceptor physiology. IEEE Transactions on Visualization and Computer Graphics (11), January 2005, p13-24.

[15] Schott, J. R. Remote Sensing: The Image Chain Approach. New York: Oxford University Press, 1997.

[16] Tsujii, O., M. T. Freedman, and S. K. Mun. Anatomic region-based dynamic range compression for chest radiographs using warping transformation of correlated distribution. IEEE Transactions on Medical Imaging (17), June 1998, p407-418.

[17] Yu, Z., W. Xiqin, and P. Yingning. New image enhancement algorithm for night vision. Proceedings of the International Conference on Image Processing (1), 1999, p201-203.

Appendix A

ENGINEERING CODE

This section contains the actual MATLAB code generated for three of the algorithms studied: APE, IMSR, and MEAM. Ancillary code that is specific to each algorithm is also included. Code for TPE is not included due to the proprietary nature of the code used during this study.

Function: APE

```
%
% APE
% Author: Seth Weith-Glushko (seth.weithglushko)
%
% Purpose:  Applies the APE algorithm to input infrared imagery
%
% Inputs:   appdata - a structure containing the application data
%           histfilter - an array containing the histogram to use in
%                        calculations
%
% Outputs:  y - an array representing the APE-enhanced infrared image
%           histfilter - an array containing an intermediate histogram

function [y,histfilter,plateau_value,first_frame]=APE(appdata),

    % Setup the parameters values from the input structure
    inputImage = floor(appdata.Processed.Data / 2^4);
    windowSize = appdata.Params.WindowSize;
    infPtFraction = appdata.Params.InfPtThreshold;
    imageWidth = appdata.Processed.Cols;
    imageHeight = appdata.Processed.Rows;
    in_maxValue = 2^12 - 1;
    out_maxValue = 2^9 - 1;
    plateau_value = appdata.Resident.PlateauValue;
    lpf_value = appdata.Params.LPFCoeff;
    first_frame = appdata.Resident.FirstFrame;

    % Find the histogram of the input image and calculate the normalized
    % CDF from it
    linData = reshape(inputImage,[1 imageWidth*imageHeight]);
    [inputHist,histValues]=hist(linData,0:2^12-1);
    unscaledCDF = cumsum(inputHist);
    CDF = unscaledCDF / max(unscaledCDF);

    % Find the indices of the greyscales needed that can be derived
    % from the CDF
    i_Min = min(find(inputHist > 0));
    i_Max = max(find(inputHist > 0));
```

```
diff = abs(CDF - 0.01);
[val,i_0P1] = min(diff);
diff = abs(CDF - 0.99);
[val,i_0P999] = min(diff);
diff = abs(CDF - 0.999);
[val,i_0P9999] = min(diff);
diff = abs(CDF - 0.25);
[val,i_A] = min(diff);
diff = abs(CDF - 0.75);
[val,i_B] = min(diff);
h_max = max(inputHist);
h_min = min(inputHist(find(inputHist)));

% Define the moving average threshold and the floored half of the
% desired window size
threshold = floor((h_max - h_min) * (infPtFraction / 100));
halfWindowSize = floor(windowSize / 2);

% Find the first inflection point. To do so, calculate the sum of a
% window centered on i_Min in the input histogram. Then, enter a loop
% and calculate the sum of a window centered one greyscale higher than
% i_Min. Calculate the difference and compare it to the threshold. If
% the difference is higher than the threshold, mark the greyscale that
% was not just iterated as the first inflection point.
prev_slice = sum(inputHist(i_Min-halfWindowSize:i_Min+halfWindowSize));
i = i_Min + 1;
while(i < i_A)
    curr_slice = sum(inputHist(i-halfWindowSize:i+halfWindowSize));
    diff = abs(curr_slice - prev_slice);
    if(diff > threshold)
        i = i - 1;
        break;
    end
    prev_slice = curr_slice;
    i = i + 1;
end
if(i == i_A)
    i_infA = i - 1;
else
    i_infA = i;
end

% Find the second inflection point. This is done in much the same way
% as the first inflection point but the moving window starts at i_Max
% and works its way back to i_B.
beginBound = i_Max - halfWindowSize;
endBound = i_Max + halfWindowSize;
if(endBound > numel(inputHist))
    endBound = numel(inputHist);
end
if(beginBound > numel(inputHist))
    endBound = numel(inputHist);
end
if(endBound < 1)
    endBound = 1;
end
if(beginBound < 1)
    beginBound = 1;
end
prev_slice = sum(inputHist(beginBound:endBound));
i = i_Max - 1;
while(i > i_B)
    beginBound = i - halfWindowSize;
```

```
        endBound = i + halfWindowSize;
        if(endBound > numel(inputHist))
            endBound = numel(inputHist);
        end
        if(beginBound > numel(inputHist))
            endBound = numel(inputHist);
        end
        if(endBound < 1)
            endBound = 1;
        end
        if(beginBound < 1)
            beginBound = 1;
        end
        curr_slice = sum(inputHist(beginBound:endBound));
        diff = abs(curr_slice - prev_slice);
        if((diff   >   threshold)   &&   (inputHist(i)   <   (0.5   *   (inputHist(i_Max)   -
inputHist(i_Min)))))
            i = i - 1;
            break;
        end
        prev_slice = curr_slice;
        i = i - 1;
    end
    if(i == i_B)
        i_infB = i + 1;
    else
        i_infB = i;
    end

    % Find the total number of pixels that come before i_infA and after
    % i_infB
    n_A = sum(inputHist(1:i_infA));
    n_B = sum(inputHist(i_infB:end));

    % Calculate the ratio value
    N = imageWidth * imageHeight;
    X = (N - (n_A + n_B)) / (n_A + n_B);

    % Calculate the nominal plateau value
    firstValue = i_0P9999 - i_infB;
    secondValue = i_infA - i_0P1;
    if(firstValue < secondValue)
        p_nom = (X * n_B) / (histValues(i_infB) - histValues(i_infA));
    else
        p_nom = (X * n_A) / (histValues(i_infB) - histValues(i_infA));
    end

    % Calculate the dynamic range of the scene
    r_d = histValues(i_0P999) - histValues(i_0P1);

    % Calculate the dynamic range factor
    if(r_d > out_maxValue)
        f_dr = 1 - (out_maxValue / r_d);
    else
        f_dr = 1 - (r_d / out_maxValue);
    end

    % Calculate the adjustment factor
    f_ed   =   1   -   ((histValues(i_A)   -   histValues(i_0P1))   /   (histValues(i_B)   -
histValues(i_0P1)));

    % Calculate the plateau parameter
    p_a = round(p_nom * f_dr * f_ed);
    % Filter the plateau value using temporal low-pass IIR filter
    if(~first_frame)
        p_a = floor(((1 - lpf_value) * plateau_value) + (lpf_value * p_a));
    end
    first_frame = 0;
    if(p_a < 1)
        p_a = 1;
    end
    plateau_value = p_a;

    % Limit the histogram using the plateau parameter
    clippedHist = min(inputHist, p_a);
    histfilter = clippedHist;

    % Using the histogram, calculate the CDF and generate a lookup
    % table to perform histogram equalization
    unscaledCDF = cumsum(clippedHist);
    CDF = unscaledCDF / max(unscaledCDF);
    LUT = zeros((in_maxValue + 1),1);
    for j=1:(in_maxValue + 1),
        LUT(j) = floor(CDF(j) * out_maxValue);
    end

    % Apply the lookup table to the contrast-enhanced image
    % and return it
    y = LUT(inputImage+1);
```

Function: MEAM

```
%
% MEAM
% Author: Seth Weith-Glushko (seth.weithglushko)
%
% Purpose:  Applies the MEAM algorithm to input infrared imagery
%
% Inputs:   appdata - a structure containing the application data
%           histfilter - an array containing the histogram to use in
%                        calculations
%
% Outputs:  y - an array representing the MEAM-enhanced infrared
%               image
%           histfilter - an array containing an intermediate histogram

function [y,histfilter]=MEAM(appdata),

    % Setup the parameters values from the input structure
    inputImage = floor(appdata.Processed.Data / 2^4);
    filterWidth = appdata.Params.FilterWidth;
    filterHeight = appdata.Params.FilterHeight;
    imageWidth = appdata.Processed.Cols;
    imageHeight = appdata.Processed.Rows;
    gainOne = appdata.Params.G1;
    gainTwo = appdata.Params.G2;
    alpha = appdata.Params.A;
    beta = appdata.Params.B;
    gainThreshold = appdata.Params.XP;
    inputMaxValue = 2^12 - 1;
    outputMaxValue = 2^9 - 1;

    % Set the output histogram to the histogram of the input image
    histfilter = hist(reshape(inputImage,[1 imageWidth*imageHeight]),0:inputMaxValue);

    % Specify an array containing one for each cell value
    % and use it as a convolution filter to find the lowpass
    % image
    averageFilter = ones(filterHeight, filterWidth);
    lowpassImage = floor((1/(filterHeight * filterWidth))*conv2(inputImage, averageFilter,
'same'));

    % Subtract the original image from the lowpass image to get
    % the highpass image
    highpassImage = inputImage - lowpassImage;

    % Take the absolute value of the image. Find the indices of
    % the image that fall above and below XP. Apply the gain values
    % G1 and G2 to those pixels
    absImage = abs(highpassImage);
    lowIndices = find(absImage < gainThreshold);
    highIndices = find(absImage >= gainThreshold);
    highpassImage(lowIndices) = floor(gainOne * highpassImage(lowIndices));
    highpassImage(highIndices) = floor(gainTwo * highpassImage(highIndices));

    % Use the linear scale algorithm to reduce the dynamic range of the
    % lowpass image
    [fl,fh,lowpassImage]        =        LinearScale(lowpassImage,alpha,beta,outputMaxValue-
30,30,imageWidth,imageHeight);

    % Add the enhanced highpass and lowpass images together
    sumImage = highpassImage + lowpassImage;

    % Limit the values of the sumImage to be between 0 and 511. Set
    % the result to y
    sumImage(find(sumImage < 0)) = 0;
    sumImage(find(sumImage > 511)) = 511;
    y = sumImage;
```

Function: IMSR

```
%
% IMSR
% Author: Seth Weith-Glushko (seth.weithglushko)
%
% Purpose:  Applies the IMSR algorithm to input infrared imagery
%
% Inputs:   appdata - a structure containing the application data
%           histfilter - an array containing an input histogram
%
% Outputs:  y - an array representing the IMSR-enhanced infrared
%               image
%           histfilter - an array containing an intermediate histogram

function [y,histfilter] = IMSR(appdata, histfilter),

    % Define certain parameters
    inputImage = floor(appdata.Processed.Data / 2^4);
    imageWidth = appdata.Processed.Cols;
    imageHeight = appdata.Processed.Rows;
    alpha = appdata.Params.Alpha;
    beta = appdata.Params.Beta;
    outputMaxValue = 2^9 - 1;
    inputMaxValue = 2^12 - 1;
    % Create a 2D array to hold the final image
    finalImage = zeros(imageHeight, imageWidth);

    % Find the FFT of the input image
    fftImage = fft2(inputImage);

    % For each field, create the Gaussian surround with the appropriate
    % weighting and then apply it to the original image. Then, subtract
    % the base ten logarithm of the lowpass filtered image with the
    % base ten logarithm of the original image. Then multiply the result by
    % the specified weighting. Finally, add this result to the final image.
    i = 1;
    while(i <= appdata.Params.NumFields)
        surround = GetSurround(imageHeight, imageWidth, appdata.Params.GausWeights(i));
        lowpassImage = fftshift(real(ifft2(fftImage .* fft2(surround))));
        tempImage = AutoGain((inputImage ./ lowpassImage), 2^15);
        [fl,fh,subtractImage]  =  LinearScale(floor(tempImage),  alpha,  beta,  511,  0,
imageWidth, imageHeight);
        modulatedImage = appdata.Params.Weights(i) * subtractImage;
        finalImage = finalImage + modulatedImage;
        i = i + 1;
    end

    % Assign the auto-gained output to y
    y = floor(AutoGain(finalImage, outputMaxValue));

end
```

Function: AutoGain

```
%
% AutoGain
% Author: Seth Weith-Glushko (seth.weithglushko)
%
% Purpose:  Applies an automatic gain to an input image
%
% Inputs:   inputImage - the 2D array to apply the gain to
%           outputMaxValue - the maximum pixel value one wishes in the output
%           image
%
% Outputs:  y - an array representing the auto-gained infrared image

function y = AutoGain(inputImage, outputMaxValue)

    % Find the minimum and maximum values of the input image
    minValue = min(min(inputImage));
    maxValue = max(max(inputImage));

    % Rescale the image into a 0-1 range and multiply it by the maximum
    % output value to get the auto-gained image
    tempImage = (inputImage - minValue) / (maxValue - minValue);
    y = outputMaxValue * tempImage;

end
```

Function: GetSurround

```
%
% GetSurround
% Author: Seth Weith-Glushko (seth.weithglushko)
%
% Purpose:  This function will generate an image representative of a
%           Gaussian function
%
% Inputs:   width - a value representing the width of the function to make
%           height - a value representing the height of the function to
%           make
%           stddev - a value specifying the standard deviation of the
%           function to make
%
% Outputs:  outputImage - a 2D array containing the scaled Gaussian function

function outputImage = GetSurround(height, width, stddev)

    % Generate an empty image to hold the Gaussian function
    gaussian = zeros(height, width);

    % For each pixel within the image, put the value of the Gaussian
    % function in the pixel.
    i = 1;
    j = 1;
    while(i <= width)
        while(j <= height)
            valueX = (i - (width / 2))^2;
            valueY = ((height / 2) - j)^2;
            topTerm = -1 * (valueX + valueY);
            botTerm = stddev ^ 2;
            gaussian(j,i) = exp(topTerm / botTerm);
            j = j + 1;
        end
        j = 1;
        i = i + 1;
    end

    % Add up every pixel within the Gaussian image and find the reciprocal.
    % Multiply that value by the entire image and return it.
    recip = 1 / sum(sum(gaussian));
    outputImage = recip * gaussian;
    outputImage = gaussian;

end
```

Function: LinearScale

```
%
% LinearScale
% Author: Seth Weith-Glushko (seth.weithglushko)
%
% Purpose:  Applies a linear scaling algorithm to an image
%
% Inputs:   inputImage - a 2D array containing an image to linearly scale
%           alpha - a value specifying a percentage of a histogram to
%                   saturate
%           beta - a value that controls how much of the image will be
%                  limited
%           width - the width of the input image
%           height - the height of the input image
%           ymax - a value that specifies the maximum digital count in the
%                  input image
%           ymin - a value that specifies the minimum digital count in the
%                  input image
%
% Outputs:  y - an array representing the linearly scaled image
%           fl - the calculated low-end gain
%           fh - the calculated high-end gain

function [fl,fh,y] = LinearScale(inputImage,alpha,beta,ymax,ymin,width,height)

    % Find the histogram of the input image
    maxValue = max(max(inputImage));
    [inputHist,histValues] = hist(reshape(inputImage,[1 width*height]),0:maxValue);

    % Find the scaled CDF of the input image
    CDF = cumsum(inputHist);
    CDF = CDF / max(CDF);

    % Find the greyscale value that corresponds to the CDF values of alpha,
    % 0.5, and 1-alpha (respectively, xa, xm, and xb)
    [val,ind] = min(abs(CDF - alpha));
    xa = histValues(ind);
    [val,ind] = min(abs(CDF - 0.5));
    xm = histValues(ind);
    [val,ind] = min(abs(CDF - (1 - alpha)));
    xb = histValues(ind);

    % Calculate all necessary intermediate values (ya, yb, ym, fl, and fh)
    ya = (ymax * beta) + (ymin * (1 - beta));
    yb = ymax - ya;
    ym = floor((ymax - ymin) / 2);
    fl = (ym - ya) / (xm - xa);
    fh = (yb - ym) / (xb - xm);
    bl = floor(ym - (fl * xm));
    bh = floor(ym - (fh * xm));

    % For each possible greyscale value, calculate the appropriate lookup
    % table, limiting values between ymin and ymax
    lut = zeros(1,maxValue+1);
    i = 1;
    while(i <= maxValue+1)
        currValue = i - 1;
        if(currValue <= xm)
            lut(1,i) = floor((fl * currValue) + bl);
        else
            lut(1,i) = floor((fh * currValue) + bh);
        end
        if(lut(1,i) <= ymin)
            lut(1,i) = ymin;
        end
        if(lut(1,i) >= ymax)
            lut(1,i) = ymax;
        end
        i = i + 1;
    end

    % Apply the LUT and save the result to y
    y = lut(inputImage + 1);

end
```

Appendix B

ALGORITHM SETTINGS

This section lists the parameters used in the generation of the test sequences shown during the psychophysical trial.

APE

- $w = 9$
- $\varepsilon = 0.01$
- $\psi = 0.3$

IMSR

- $S = 3$
- $W_s = \{0.33, 0.33, 0.33\}$
- $\sigma_s^2 = \{4, 40, 200\}$
- $\alpha = 0.01$
- $\beta = 0$

MEAM

- Filter size: 3x3 pixels

- $g_1 = 10$
- $g_2 = 0.5$
- $x_p = 5$
- $\alpha = 0.02$
- $\beta = 0$