Design of Region Merging Algorithm for Segmentation of Noisy Images

Darcie Dillon, Katharine Kerr, and Sungbok Lee
Supervisors: Fady Alajaji and Abdol-Reza Mansouri

Mathematics and Engineering
Queen's University
Kingston, Ontario
Canada

April 7, 2014

Acknowledgements

We would like to thank Dr. Linder for organizing MTHE493 and Johana Ng for sending out helpful e-mails and organizing the presentations.

Most importantly, we would like to thank Dr. Alajaji and Dr. Mansouri for their guidance and help throughout the term. Their advice for improvements in design and help with understanding the project is greatly appreciated.

Abstract

The region merging technique is a method for implementing image segmentation. Two image models were implemented to evaluate the merging criterion as discussed by Felipe Calderero and Ferran Marques in “Region Merging Techniques Using Information Theory Statistical Measures”. The two image models are the i.i.d. model and the Markov process model. These two image models allow us to mathematically analyze an image and hence implement a region merging algorithm. The merging algorithm is evaluated both qualitatively, by observing the segmented image, and quantitatively by calculating the PSNR value between the initial and merged images. Two different types of image noise, salt and pepper and additive white Gaussian noise, were applied to images to assess the robustness of our merging algorithm. The results demonstrated that without noise, the Markov model performed superiorly to the i.i.d. model for meaningful image segmentation. In addition, the algorithm was fairly robust to salt and pepper noise for both image models, where the application of a median filter improved results. If additional number of filters were applied, the image became under-segmented. For additive white Gaussian noise, the i.i.d. model performed significantly worse than the Markov model; however, the i.i.d. results were improved through the use of median filters and the Markov case was not.

Contents

1

Introduction

Ideas that only belonged in science fiction are becoming a reality through advancements in the field of computer vision. Computer vision provides methods for processing and analyzing images to enable machines to view images the way humans do. One such method for analyzing images is image segmentation. Image segmentation involves partitioning an image into its meaningful parts so that it can be processed by a computer.

One real-life application of image processing is the Google self-driven car [1]. The car is capable of driving to a destination without accident. It does this by generating a 3D map of its surrounding environment and responding to this input. Another application is a defect inspection system by KLA Tencor that can detect defects in products by using computer vision and optics [2].

Through the use of image segmentation, images can be processed and analyzed by computers in fields such as manufacturing, autonomous driving and graphic design. One successful approach to image segmentation is the region merging technique. As the name suggests, it initially splits the image into small partitions, usually no bigger than 15x15 pixels. Then a region merging algorithm is applied to the partitions to iteratively merge them into the meaningful regions of the image. Our goal is to design a region merging algorithm that can be used on noisy images to achieve a meaningful segmentation.

1.1 Motivation

The applications of computer vision can improve quality of human life. The examples below show how improvement would be possible by increasing safety, accuracy of analyzing images and simplifying difficult tasks. Along with the benefits of computer vision, specifically image segmentation, there would be negative impacts on society, environment and economy, as discussed in Section 1.2.

1.1.1 Applications

All following applications rely heavily on computer vision and many of the following examples can be implemented with the region merging technique.

Manufacturing Systems
Machine vision could improve the efficiency of automated manufacturing systems in terms of speed and the number of different tasks the machine is able to perform. For example, a machine could be able to perform quality control, alerting workers when manufactured parts have defects, or are not within tolerances. Machines could also continuously monitor equipment and alert maintenance crews when stresses could become failures. This would decrease the number of malfunctions of machines, which in turn would reduce the number of employee accidents.

Autonomous Driving
Image processing could be used in autonomous driving for surveying the surrounding environment, such as the Google self-driven car [1]. The car could automatically make decisions based off of its environment and as such avoid accidents caused by human error. This would greatly decrease the number of accidents and hence fatalities and injuries as cars could react to avoid objects when humans do not.

Graphic Design
Image processing could reduce the time associated with various graphic design tasks. The ability for image segmentation to determine the outline of the important objects in an image could be used to cut and paste objects from images. For example, iChat allows for users to change the background of a videoconference through image processing. The design and use of an image segmentation algorithm could allow more graphic design processes to become automated [3].

Mining
In open pit mines, shovels are used to transport rocks. If a missing shovel tooth is not detected early it can lead to other avoidable problems. Image processing can be used to detect missing shovel teeth and then alert maintenance workers of the problem. This would result in greater productivity for the mine and reduce the down time of shovels [4].

Medical Imaging
Computer vision could help doctors make more accurate diagnoses. Image processing could provide a means for detecting small or hard to distinguish areas of concern within medical images. Also, it could decrease the amount of time a doctor needs to spend analyzing an

image for diagnosis. If the image processing was equipped to handle noise, a computer might be able to distinguish or recover important objects in the image. This would lead to more accurate diagnoses and thus better treatments for patients.

1.2 Impact of a Solution

Image segmentation could have a positive impact on society by improving the quality of life. As mentioned above, computers could analyze medical images, which could increase the accuracy of diagnosis. Accidents, both vehicular and in the workplace, could be decreased with the use of machines that react to visual input to avoid accidents caused by human error. Also, the negative societal affects of driving under the influence could be reduced, if automated cars could accurately drive themselves to a given location. Computer vision could positively impact the environment as autonomous robots could be used to clean up garbage in cities and rural areas by being able to distinguish between garbage and other objects. Computer vision could help the economy by providing workers with more maintenance jobs for computer vision machines. An industry could be created for building, maintaining and researching improvements in the field of computer vision.

As computers with machine vision become more prevalent in society, the amount of garbage and waste associated with these machines will increase. In addition, as the demand for these machines increases, so will the demand for the materials used in the fabrication of these machines. This will negatively affect the environment, as more machines will be discarded and more resources consumed. Lastly, computer vision could have a negative effect on the economy as more menial jobs in manufacturing could be replaced by machines. However, this could be offset by the increase in jobs associated with building, maintaining and researching improvements for these machines. This in turn would require people to have higher education, thus increasing the number of qualified workers but decreasing the availability of menial jobs.

1.3 Problem Definition

The objective of our project is to design a region merging algorithm to partition an image into its meaningful segments. Specifically, we have reduced the problem to working only with greyscale images. The algorithm will be evaluated for use with noisy images, either with salt and pepper noise or additive white Gaussian noise. We will evaluate the robustness of our merging algorithm to noise and the results of applying a filter to noisy images before image segmentation.

2

Background

Before the design of our algorithm can begin, a few fundamental results from probability and information theory need to be understood. These results form the basis on which our algorithm was built. The algorithm decides which two regions to merge by evaluating the probability that these two regions are from the same object in the image. Since computers can only distinguish different objects within an image through evaluating the pixel values, the algorithm must work strictly with the image as an array of pixel values. A method for distinguishing different regions in an image is to make comparisons between the true underlying probability distributions of the respective regions. Thus, we consider the original image as a stochastic process and use information theory metrics to decide which regions in the image should be merged.

2.1 Quantization

Quantization is used to reduce the complexity of the greyscale image. As greyscale images contain a finite number of pixel values, 256, only discrete level quantizers were studied. For the algorithm, quantization is the process of mapping values from a discrete set, $\mathcal{X}$, to a discrete set, $\mathcal{C}$

$$Q : \mathcal{X} \longrightarrow \mathcal{C}$$

where $|\mathcal{X}| > |\mathcal{C}|$. $\mathcal{X}$ is uniformly partitioned into b-many bins, B_i where $i = \{1, ..., b\}$. Then,

$$\bigcup_{i=1}^{b} B_i = \mathcal{X}$$

The quantizer for our design can formally be defined as a b-point uniform quantizer where $\mathcal{C} = \{y_1, ..., y_b\} \subset \mathcal{X}$ and each y_i is the floor of the midpoint of the bins B_i where $i = \{1, ..., b\}$.

2.2 Random Processes

A random process, or stochastic process, is used to model the pixels in the image. In probability theory, a stochastic process is a collection of random variables; this is often used to represent the evolution of some random value, or system, over time. However, in our project, each pixel value is considered as a random variable. For the precise mathematical definition, a stochastic process X is a collection

$$\{X_t : t \in \mathbb{Z}_{\geq 0}\}$$

where each $X_t \in \mathcal{X}$.

2.2.1 i.i.d.

A sequence, or collection of random variables, is independent and identically distributed (i.i.d.) if every random variable has an identical probability distribution and are all mutually independent. In other words, for each $X_t \in \{X_t : t \in \mathbb{Z}_{\geq 0}\}$

$$P(X_i = a) = P(X_j = a)\ \forall i, j \in \mathbb{Z}_{\geq 0}, \forall a \in \mathcal{X}$$

2.2.2 Markov

A Markov process is a stochastic process that satisfies the Markov property:

$$P(X_i = a_i | X_{i-1} = a_{i-1}, ..., X_0 = a_0) = P(X_i = a_i | X_{i-1} = a_{i-1})\ a_i \in \mathcal{X} \text{ and } \forall i \in \mathbb{Z}_{\geq 0}$$

The Markov property implies that the next state depends only on the present state and is conditionally independent from the other states.

2.3 Information Theoretic Measures

2.3.1 Entropy

Entropy is a measure of the uncertainty of a random variable. Let X be a discrete random variable with alphabet $\mathcal{X}$ and probability mass function $p(x) = P(X = x), x \in \mathcal{X}$.
The entropy $H(X)$ is defined mathematically as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \cdot \log_2 p(x)$$

2.3.2 Divergence

Divergence, or relative entropy, $D(p\|q)$, is a measure of the distance between two probability mass functions p and q. It is mathematically defined as

$$D(p\|q) = \sum_{x \in \mathcal{X}} p(x) \cdot log_2 \left(\frac{p(x)}{q(x)} \right)$$

2.3.3 Log-Likelihood Ratio

Let $\{x_1, x_2, ..., x_n\}$ be a sequence of observations. Then construct two hypotheses on the underlying source distribution:

1. H_0: The "null hypothesis" where the sequence has distribution P_{X^n}
2. H_1: The "alternative hypothesis" where the sequence has distribution $P_{\tilde{X}^n}$

We define the *likelihood ratio* for the two hypotheses as:

$$L := \frac{P_{X_n}(x_1,x_2,...,x_n)}{P_{\tilde{X_n}}(\tilde{x}_1,\tilde{x}_2,...,\tilde{x}_n)}$$

The *log-likelihood ratio* is simply the log of the above function:

$$log_2(L) := log_2 \left(\frac{P_{X_n}(x_1,x_2,...,x_n)}{P_{\tilde{X_n}}(\tilde{x}_1,\tilde{x}_2,...,\tilde{x}_n)} \right)$$

2.3.4 Neyman-Pearson Lemma

The *Neyman-Pearson lemma* states that the optimum test for the comparison of two hypotheses is the likelihood ratio test.

2.4 Related Research

The design of our algorithm was based off of the ideas on image segmentation through region merging in Calderero and Marques' paper: "Region Merging Techniques Using Information Theory Statistical Measures" [5]. The paper discusses three different stochastic process in which to model an image: i.i.d., Markov chain and Markov random field. In addition, it discusses two different types of merging criteria: Kullback-Leiber and Bhattacharyya.

For the design of our algorithm, we chose to work with two of the image models and one merging criterion: i.i.d., Markov chain and Kullback-Leiber respectively. The limits placed on the number of different models and merging criterion were due to time constraints.

Our project was also an extension of two previous fourth year thesis projects, "Design of Region Merging Algorithms for Image Segmentation" [6] and "Design of Region Merging Algorithms for Image Segmentation with an Extension to Splitting" [7]. We expanded on their projects by choosing to implement the algorithm for use with noisy images.

3

Design

The first step in the design of our region merging algorithm is modelling the image. For our project, we have chosen to model the image as an i.i.d. and a Markov chain process. Once the image has been described mathematically, there exits a method for quantitatively analyzing whether two regions within an image should be merged. The image is partitioned and then these regions are iteratively merged using this method until a stopping criterion is met. In this section, we will describe in detail the steps involved in the design and implementation of our region merging algorithm.

3.1 Modelling an Image

The very first step in mathematically modelling the image is to convert the image into its set of pixel values. Since we are only working with greyscale images, each pixel will take its value from the set $\mathcal{X} = \{1,2,...,255\}$. The image will be quantized to reduce the size of the set of pixel values. Refer to section 2.1.

After the image is converted into its set of pixel values, it can be considered as a two-dimensional matrix and a corresponding two-dimensional array can be constructed. Then the array is partitioned into a number of equal sized subarrays, where each of these subarrays will represent an initial partition. For our algorithm, the initially partitioned array is considered as a mutually independent random process, which is defined as a region. The region merging algorithm merges the two adjacent regions with the highest likelihood of being from the same object. Hence, we need to define what it means to be adjacent. We will employ the definition of adjacency as follows.

Consider two regions R_i and R_j and let r_i and r_j be the index for each subarray corresponding to these regions. Two regions, R_i and R_j, are formally declared to be adjacent if for some index pair $(k, l) \in r_i$, at least one of $(k, l+1), (k, l-1), (k+1, l)$, or $(k-1, l) \in r_j$ [7]. Region merging will only be done between two regions that satisfying the definition of adjacency.

Two different mathematical models for images were implemented in our algorithm. The first one is the i.i.d model and the second one is the Markov chain. In the i.i.d. model, the value of each pixel is considered as an independent and identically distributed random variable; whereas, in the Markov chain model, the value of each pixel depends on its neighbouring pixels. The Markov chain model will represent the local homogeneity of each region better than the i.i.d case, and hence should produce better results.

3.1.1 i.i.d.

The most important idea of i.i.d. modeling is the assumption that there is no dependency between two arbitrary pixel values. Below we discuss how the true probability distribution of each region in the image can be approximated by the empirical distribution and how the merging cost equation for the merging algorithm is derived.

Type of Region
Let R be any arbitrary region of the image. Then we can define the empirical distribution (or type) of a region as follows:

$$P_R(x) = \tfrac{N(x|R)}{n}$$

where x is the value of the pixel from the set $\mathcal{X}$, $N(x|R)$ is the number of x occurances in region R and n is the total number of pixels in region R. The type of a region can approximate the true underlying probability distribution of the region as explained below.

From Calderero and Marques paper, the probability of the type of a sequence of i.i.d. observations R with probability distribution Q is given by

$$Q^n(R) = 2^{-n(H(P_R)+D(P_R\|Q))}$$

where $H(P_R)$ is the entropy of pixels in region R, $D(P_R\|Q)$ is the relative entropy between the type and true probability distribution and n is the number of pixels in the region. From Calderero and Marques paper, it is stated that type P_R converges to Q, concretely, $D(P_R\|Q) \to 0$ with probability 1 as $n \to \infty$. Hence, for n sufficiently large, we can approximate the true probability distribution Q as

$$Q^n(R) = 2^{-n(H(P_R))} \tag{3.1}$$

where $Q^n(R)$ only depends on $H(P_R)$. This states that the unknown true distribution of the data $Q^n(R)$ can be well approximated by type P_R given n is large enough.

Derivation of Merging Cost Equation
Now recall the two different hypotheses from Chapter 2. In order to determine which two neighbouring regions should be merged, the following two hypotheses from Calderero and Marques are considered [5]:

- H_0: pixels in the first region, $\mathbf{x_1} \in R_1$, and pixels in the second region, $\mathbf{x_2} \in R_2$, are both distributed by $P_{R_{1\cup 2}}$;
- H_1: pixels $\mathbf{x_1} \in R_1$ are distributed by P_{R_1}; and pixels $\mathbf{x_2} \in R_2$ are distributed by P_{R_2}.

Then the Neyman-Pearson lemma is applied which states that the log-likelihood ratio test is the optimum test for these hypothesis. The log-likelihood ratio test for these hypotheses can be written as

$$log_2 \frac{P_{H_0}(R_1, R_2)}{P_{H_1}(R_1, R_2)} = -(n_1 + n_2) \cdot H(P_{R_1 \cup R_2}) + n_1 \cdot H(P_{R_1}) + n_2 H(P_{R_2}) \tag{3.2}$$

where

$$P_{R_1 \cup R_2}(a) = \frac{n_1}{n_1 + n_2} \cdot P_{R_1}(a) + \frac{n_2}{n_1 + n_2} \cdot P_{R_2}(a) \tag{3.3}$$

and n_1 and n_2 denote the number of pixels in regions R_1 and R_2, respectively. If we apply the Kullback-Leibler divergence between two statistical distributions with considering equation (3.3), we can rewrite (3.2) as

$$log_2 \frac{P_{H_0}(R_1, R_2)}{P_{H_1}(R_1, R_2)} = n_1 \cdot D(P_1 \| P_{R_1 \cup R_2}) + n_2 \cdot D(P_2 \| P_{R_1 \cup R_2}) \tag{3.4}$$

This will be the final equation for the merging cost between two adjacent regions.

3.1.2 Markov Chain

The second model assumes that pixel values have the Markov property. The value of each pixel will be dependent on the values of its neighbouring pixels. These neighbouring pixels include the pixels on right, left, top and bottom of the current pixel. Therefore, we can consider four Markov chains, which have the directions moving toward the left, right, top or bottom.

For each direction, the empirical co-occurrence matrix can be constructed to represent the empirical distribution of the region. This probability co-occurrence matrix will have all transition probabilities between every pixel value. Therefore, if each pixel takes a value from a set $\mathcal{X}$, the matrix will have size $|\mathcal{X}| \times |\mathcal{X}|$. Each transition probability from value

a to b can be defined as follow

$$P_R(a,b) = \frac{N(X_{(i,j)} = a, X_{(i,j)+\Delta} = b|R)}{N_D} \tag{3.5}$$

where $\Delta \in \{(1,0),(0,1),(-1,0),(0,-1)\}$ is the direction of the Markov chain,

$$N(X_{(i,j)} = a, X_{(i,j)+\Delta} = b|R)$$

is the number of transition from pixel value a to b to the direction Δ within the region R and N_D is the total number of transition to the direction Δ in R. We will assume that this Markov process is ergodic, and hence, it is completely characterized by its initial state and the probability co-occurrence matrix. Since the pixels in the region are not ordered, we can assume that all initial states are equally likely. Thus the probability of the pixel value i is initially set to $\pi_i = \frac{1}{|\mathcal{X}|}$. For simplicity, the algorithm implemented Markov chain that has direction to the right.

Derivation of Merging Cost Equation
Assume that region R has n number of pixels. Then, the region R can be considered a Markov process having n sequences. Let this sequence be $\mathbf{x} = \{x_1, x_2, ..., x_n\}$. Then with the assumption that the Markov process is ergodic,

$$P_R(\mathbf{x}) = P_R(x_1) \prod_{i=2}^{n} P_R(x_i|x_{i-1}) \tag{3.6}$$

which defines the joint probability distribution of the sequence $\mathbf{x}$ in R. Consider the same two hypotheses from the i.i.d. case, the log-likelihood ratio test can be written by using equation 3.6 as

$$\begin{aligned} log_2 \frac{P_{H_0}(\mathbf{x})}{P_{H_1}(\mathbf{x})} &= log_2 \frac{P_{R_1 \cup R_2}(x_1)}{P_{R_1}(x_1) P_{R_2}(x_1)} - (n_1 + n_2 - 1) \cdot H(P_{R_1 \cup R_2}) \\ &+ (n_1 - 1) \cdot H(P_{R_1}) + (n_2 - 1) \cdot H(P_{R_2}) \\ &\approx -(n_1 + n_2) \cdot H(P_{R_1 \cup R_2}) + n_1 \cdot H(P_{R_1}) + n_2 \cdot H(P_{R_2}) \\ &\propto -n_1 \cdot D'(P_{R_1} \| P_{R_1 \cup R_2}) - n_2 \cdot D'(P_{R_1} \| P_{R_2 \cup R_2}) \end{aligned} \tag{3.7}$$

where $$D'(P_i || P_{i \cup j}) = \sum_{i,j \in \mathcal{X}} \pi_i \cdot p_{ij} \cdot \log_2 \frac{p_{ij}}{q_{ij}}$$

This equation will be used to calculate the merging cost for the Markov model algorithm.

3.2 Algorithm

3.2.1 Quantization

For the algorithm, only greyscale images and a uniform quantizer were used for simplicity. Greyscale images greatly reduces the complexity of the algorithm. In addition, the results would not be affected by using only greyscale images, as the results are comparative between two greyscale images: the quantized image and the merged image. Recall, that for greyscale images pixels take values in the alphabet:

$$\mathcal{X} = \{0, 1, ..., 255\}$$

Quantization maps $\mathcal{X}$ to a smaller set $\mathcal{C} = \{y_1, ..., y_b\}$, where b is the number of bins in the uniform quantization and y_i is as defined in Section 2.1. The number of quantization bins is a parameter in the program, which was varied to produce different results. It was found that choosing 10-20 bins allowed for an accurate representation of the image for the merging algorithm.

3.2.2 Partitioning

Before merging begins, the image must be partitioned into initial partitions. Two main choices concerning the initial partition must be addressed: the partition shape and size.

A square shape was chosen for the initial partitions due to its simplicity and to reduce the complexity of the algorithm. The trade-off between implementing a more complicated shape, such as the watershed algorithm used by Calderero and Marques [5], is not time effective and as such only a square initial partition shape was chosen.

Let us assume an initial partition as a square of $n \times n$ pixels and the image as an array of $m_1 \times m_2$ pixels. Then define:

$$r_1 \equiv m_1 \bmod n \text{ and } r_2 \equiv m_2 \bmod n$$

where r_1 and r_2 represent remaining pixels that need to be included in a partition. To include these remaining pixels in the initial partitions, it was chosen that the right edge initial partitions be of size $(n + r_1) \times n$ and the bottom edge initial partitions be of size $n \times (n + r_2)$.

Using a square initial partition allows for the size of the partition to be chosen. The size of the initial partitions has a strong effect on the resources required for the execution of the algorithm. Since the initial partitions are iteratively merged to form a final partitioned image, an increase in the number of initial partitions or regions results in an exponential increase in the number of merges of the algorithm.

As mentioned previously in section 3.1.1, the initial partition must be large enough such that the empirical distribution accurately represents the true underlying distribution of the pixels in the partition. Conversely, if the initial partition is too large then multiple semantic regions could be contained in one initial partition. The initial size of the partition for each image was chosen using trial and error.

3.2.3 Merging

After the image has been quantized and partitioned into its smallest elements, these elements need to be iteratively merged to output the semantically meaningful partitioned image. For each iterative merge, two questions need to be answered:

1. Which two regions should be merged next?
2. How should these regions be merged?

A region can only be merged with one of its neighbouring regions. A neighbouring region is defined as being either above, below, to the right or left of the region in question.

To answer the two questions defined above, two neighbouring regions R_i and R_j, with empirical distributions, P_{R_i} and P_{R_2}, respectively, are defined. The pixels in the first and second regions are defined as $\mathbf{x_i} \in R_i$ and $\mathbf{x_j} \in R_j$.

As mentioned in section 2.3.4, the Neyman-Pearson lemma proves that the optimal test for two hypotheses is the likelihood ratio test. As derived in section 3.1.2, the log-likelihood ratio for H_0 and H_1 can be written as:

$$\log \frac{P_{H_0}(\mathbf{x_i}, \mathbf{x_j})}{P_{H_1}(\mathbf{x_i}, \mathbf{x_j})} = n_i \cdot D(P_{R_i} \| P_{R_{i \cup j}}) + n_j \cdot D(P_{R_j} \| P_{R_{i \cup j}})$$

where the probability distributions P_{R_i} and P_{R_j} are the empirical distribution of R_i and R_j, respectively for the i.i.d. model and the empirical probability transition matrix of R_i and R_j, respectively for the Markov model.

$P_{R_{i \cup j}}$ is calculated as follows:

$$P_{R_{i \cup j}} = \tfrac{n_i}{n_i + n_j} \cdot P_{R_i} + \tfrac{n_j}{n_i + n_j} \cdot P_{R_j}$$

A larger log-likelihood ratio correlates to a stronger likelihood that regions R_i and R_j should be merged. In terms of the merging algorithm, the log-likelihood ratio measures how likely it is that two regions should be merged and is used as the cost function. However, for implementation the log-likelihood ratio is multiplied by -1 so that a smaller cost relates to a higher probability that R_i and R_j are distributed by $P_{R_{i\cup j}}$. The cost function is as follows for R_i and R_j:

$$Cost(R_i, R_j) := n_i \cdot D(P_{R_i}||P_{R_{i\cup j}}) + n_j \cdot D(P_{R_j}||P_{R_{i\cup j}})$$

At each iteration the algorithm merges the two neighbouring regions with the lowest cost. The new merged region will have the probability distribution $P_{R_{i\cup j}}$ as defined above.

3.2.4 Stopping Criterion

The stopping criterion stops the algorithm from continuously merging regions. Two stopping conditions were used such that the resulting image would be partitioned into its meaningful segmentations. The algorithm will stop by either reaching a merging cost limit or a minimum number of remaining regions. The maximum merging cost was varied and the minimum number of regions remaining was set to 3.

3.3 Data Structures

For the implementation of the region merging algorithm, unique data structures were defined to store and access related information. Data structures helped organize and simplify the implementation of the merging algorithm as a data structure can be accessed with a single pointer. The data structures used were strongly influenced by the data structures defined in last years thesis [7].

The data structures form a hierarchy of components that describe the different elements of the merging algorithm. These components and their relationships are described in this section.

3.3.1 Atoms

Before merging occurs, the image is subdivided into initial partitions. An atom in the program is an initial partition of the image. It is the smallest element of the image and will be iteratively merged to form regions. Each atom will be represented as an arry of pixel values of size $n \times m$. The majority of the atomss will have size $n \times n$ except if the atom is at the edge. If it is, it will take on the remaining pixels as well. The size of this $n \times n$ array is determined by the initial partition size.

Each atom contains identifying information about the pixel values within itself in order to calculate the associated merging cost. Only one copy of the quantized image pixel array is kept in memory in order to greatly reduce memory resources and execution time.

Therefore, an atom needs to contain information that identifies where its pixels lie within an image. For this, an atom stores its x-coordinate, y-coordinate, width n and height m, where the x and y coordinates identify the position of the top-left hand corner pixel of the atom.

As atoms are merged together, they form regions. The program groups atoms into regions through a linked list, where each link contains a pointer to the memory location of the next atom within the region. Also, the program needs to keep track of which region an atom belongs. This is done through a region identification (ID) parameter, which is stored with each atom.

3.3.2 Regions

A region is a group of merged atoms. These regions segment the image into partitions, where the goal is to have the final regions accurately representing the important elements of the image. For every merging iteration of the program, two regions are merged together to form a new region. The region with the smaller region ID is kept and the other region is erased from memory. The merging of two regions requires the two linked lists of atoms to be combined. The atom linked lists are merged by assigning the head of one linked list at the tail of the other. Therefore, each region needs to keep track of the memory location of the head atom in its atom-linked list.

As mentioned in section 3.2.3, a cost function is used to determine which two regions should be merged next. To calculate the cost function, a region needs to keep track of its size and empirical distribution. To reduce the time associated with finding the next best merge within the image, a region keeps track of which neighbouring region will have the lowest cost associated with a merge and the cost of that merge. Regions can only merge with their neighbours as defined in section 3.2.2. Therefore, each region contains the head pointer to a linked list of its neighbours.

3.3.3 Neighbours

Having a data structure for neighbours greatly reduces the time associated with finding the current neighbours of a particular region. Neighbours of a region are grouped together and stored as a linked list. Each neighbour contains a pointer to the next neighbour in the linked list. Each neighbour contains the region ID of neighbouring region and the cost associated with merging with that neighbouring region.

3.3.4 Image

The image is the largest element of the program. To get a sense of the hierarchy within the data structures, the image is partitioned into regions, where each region has associated neighbours for merging and a list of atoms within that region. The image data structure links all the elements of the program together. It contains an array of the current regions within the image. In addition, it contains information of the next best regions to merge over the whole image. This is stored as a region ID of one of the best regions to merge. Lastly, the image contains its pixel height and width.

3.4 Noise

Image noise is the random alteration of pixel values during image capture or transmission, which results in changes to the underlying distribution of pixels in regions. Our merging algorithm design uses these distributions to determine which regions should be merged. Thus, the presence of noise affects the efficacy of our algorithm, specifically by causing two types of segmentation errors: over-segmentation and under-segmentation. Over-segmentation occurs when semantic regions are unduly merged. Conversely, a lack of region merging results in under-segmentation. The effects of two types of image noise were considered: salt and pepper noise, and additive white Gaussian noise.

3.4.1 Salt and Pepper Noise

Salt and pepper noise randomly replaces pixels in the image with either a white or black pixel, with pixel values 0 and 255 respectively. Typically the black pixels will be in light regions and the white pixels in dark regions. The percentage of pixels affected by the noise can be controlled and was studied from 2%-30%.

3.4.2 Additive White Gaussian Noise

Additive white Gaussian noise affects the image array of pixels, X, by adding an independently generated random variable, Z, with zero mean Gaussian distribution, $Z \sim N(0, \sigma)$.

$$P(Z) = \tfrac{1}{\sigma\sqrt{2\pi}} e^{(-\frac{z}{2\sigma^2})}$$

where $P(Z)$ is the probability density function of a Gaussian random variable Z and σ is the standard deviation. First, X is normalized to the interval $[0, 1]$. Let X' be the normalized image. Then, the output of the image with Additive white Gaussian noise, Y

is defined as:

$$Y_i = X_i' + Z_i$$

where i is the ith pixel in the image. Variance was varied between 0.01 and 0.3.

Median Filter
A median filter was applied to noisy images to try and recover the original image. A median filter works by running through the image pixel by pixel, starting at the top-left pixel and moving right, replacing the pixel with the median value of its neighbouring pixels. The neighbouring pixels are considered to be within the window of the filter [8]. For our algorithm, we chose a 3x3 box around the pixel in question.

Using a filter partially removes the noise from the image and as such changes the distribution of the pixels within a region to a closer approximation of the original image. The median filter is not susceptible to outlying pixels as it uses the median value of pixels within the window. This makes it a great choice for salt and pepper noise, as salt and pepper noise changes pixels to extreme values. In addition, a median filter cause relatively less blurring of the edges in an image [9].

3.5 Evaluation

To evaluate the results of our algorithm, we assessed the images both quantitatively and qualitatively. Both methods were used as a good quantitative analysis may not imply a good qualitative result.

3.5.1 Evaluating Results

Quantitative
To quantitatively evaluate the performance of our algorithm, the peak-to-noise ratio (PSNR) was calculated between the quantized and the segmented image. PSNR is defined mathematically as

$$PSNR = 20 \cdot \log_{10}(\tfrac{255}{\sqrt{MSE}})$$

where MSE is the mean squared error between the quantized image and the merged image. A low MSE is ideal and clearly, a low MSE results in a high PSNR value. Thus, a high PSNR value is desired when evaluating the merged images.

It is important to note that a high PSNR does not imply a well segmented image. In the ideal case, the MSE between the quantized image and the segmented image would be zero;

however, this would imply that the pixel values have not changed through merging. The pixel values of the regions are calculated by averaging the original quantized image pixels within that region. Therefore, a higher PSNR is directly related to the number of regions that remain. The more regions that remain after merging, the higher the PSNR. PSNR cannot be used as the only metric to measure the performance of our algorithm. However, it will be a useful measure for comparing two similar segmented images.

Qualitative Evaluation
Our goal is to have the algorithm to segment the image into the regions that are meaningful to humans. Take, for example, a tree on a hill with clouds in the sky. It would be desired to have three regions left in the merged image: the tree, the hill, and the sky. If the image is not iteratively merged enough, there may be multiple regions within each object and the image is considered under-segmented. If the image is iteratively merged too many times, the tree and hill may become one region and the image is over-segmented. Both of these cases are undesirable. Thus it is important to visually inspect the merged image for the regions it contains and evaluate if these are the meaningful regions within the image.

3.5.2 Evaluating Efficiency

The algorithms efficiency is evaluated on the amount of required resources for execution. The two main resources that the algorithm is evaluated on are execution time and required memory. The application of the region merging algorithm changes the restrictions placed on these resources. In a medical application, accuracy is very important and could be preferred over speed and memory. However, in an application like the Google car, images need to be processed in as close to real time as possible to avoid accidents.

The execution time and required memory for the program is directly related to the complexity of the algorithm. If the program is given more initial partitions and a greater number of quantization bins, then the complexity of the algorithm increases and the amount of time and memory required for execution increases. An algorithms complexity is evaluated by the number of steps required for output given an input size, n. This is written in Big O notation. For the merging algorithm, the input size, n, is correlated to the size of the initial partitions and the number of quantization bins.

3.6 Language Selection

The region merging algorithm was implemented with two programming languages: MATLAB and C. Each language was carefully chosen based off of the unique functionality that they provide.

MATLAB was chosen due to its predetermined functions and our familiarity with the programming language. MATLAB was used for the quantization of images, changing an image from its .jpg form to an array of pixels and vice versa, adding noise to an image and removing the noise via a median filter. MATLAB simplified the quantization of an image since the coding structures were familiar. In addition, for image input and output, MATLAB has very efficient predefined functions. Similarly for adding and removing noise, there are predefined functions for the two types of noise we studied, salt and pepper noise and additive white Gaussian noise with pre-constructed median filter.

The programming language C was chosen to implement the partitioning and region merging algorithm. C is a low level language and as such offers more flexibility when writing complex programs. C allows for unique data types to be defined and assigned memory as a single unit. The ability for related information to be stored and accessed through a single pointer greatly reduced the complexity of programming the partitioning and merging algorithm. C was used to define data structures for atoms, regions, neighbours and the overall image. The information stored within each of these structures is as defined in the previous section.

Another benefit to C is that it allows for memory allocation. Memory can be assigned and released as needed and thus reduces the amount of memory required for the merging algorithm to run. For example, with each merge of the algorithm, the two regions R_i and R_j are merged into R_i and the memory for R_j is released. Lastly, the execution time of the merging algorithm is greatly reduced by programming in C over MATLAB. Mainly, this is due to the reasons mentioned above: the increased flexibility in programming and the control of memory allocation C provides.

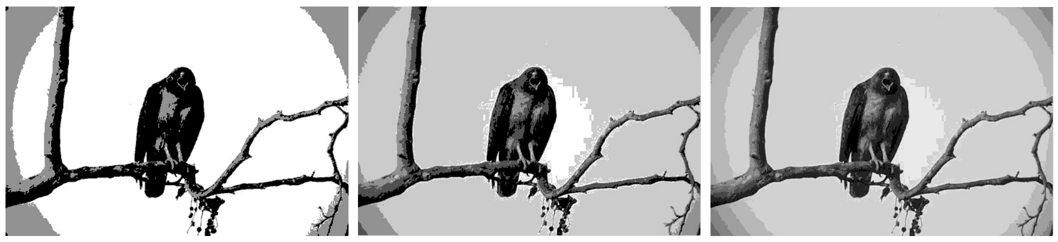
In addition C was chosen for calculating the quantitative evaluation, the PSNR between the quantized image and merged image. C was chosen for simplicity, as the execution costs of implementing it in MATLAB would have been negligible.

4

Results

The merging algorithm was tested with three images from two different types of images; animal and landscape. The three images used for visual results are of a bird on a branch (image 1), rocks with hills in the background (image 2) and an island in a body of water (image 3).

Figure 4.1 shows image 1 with 3, 5 and 10 bin quantization.



(a) 3 bins *(b) 5 bins* *(c) 10 bins*

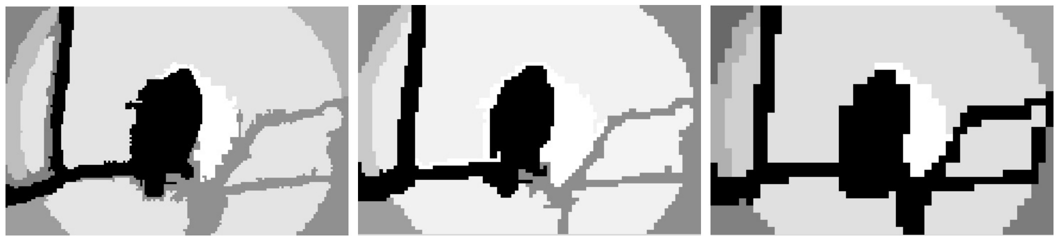
Figure 4.1: Image 1 with 3, 5, and 10 bin uniform quantization.

As shown above, 10 bin quantization gives an accurate representation of image 1. All images in the rest of the results are with a 10 bin uniform quantizer and a 5x5 initial partition size, unless otherwise stated. Image 1 is merged with a maximum merging cost of 5000, and images 2 and 3 are merged with a maximum merging cost of 7000.

4.1 Initial Partition Size Evaluation

The initial partition size was varied to evaluate the effect that it has on our algorithm. The chosen sizes were 3x3, 5x5, 10x10, and 15x15.

Figure 4.2 shows image 1 with various initial partition sizes.



(a) 3x3 initial partition size *(b) 5x5 initial partition size* *(c) 10x10 initial partition size*

Figure 4.2: Image 1 merged with 3x3, 5x5 and 10x10 initial partition size.

Figure 4.3 shows the performance of different initial partition sizes evaluated for PSNR versus various quantization bins.

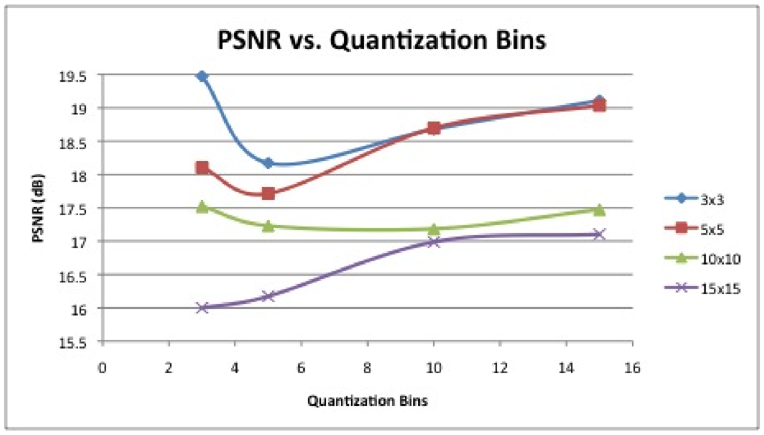
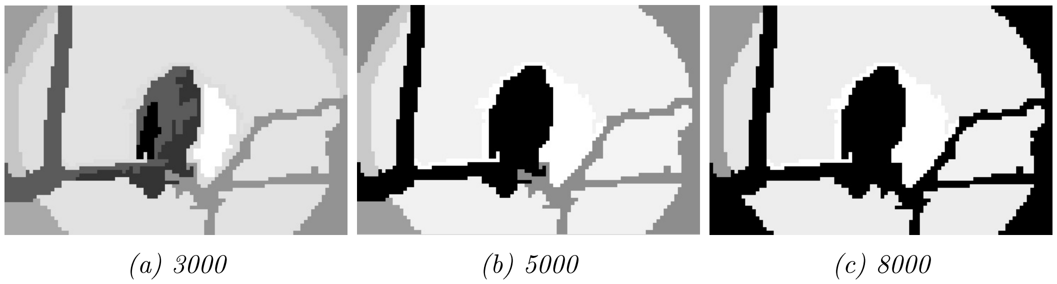


Figure 4.3: PSNR versus number of quantization bins for image 1 merged with different initial partition sizes for the i.i.d. model.

Figure 4.2 shows that qualitatively the 5x5 initial partition size yields the best result as the bird is more accurately captured as one object. Figure 4.3 shows the PSNR value for the 3x3 initial partition size versus the 5x5 initial partition size is negligible after 10 quantization bins. Thus, Figure 4.2 (b) would be chosen as the most semantically merged image. Hence, an initial partition size of 5x5 will be continuously used to discuss rest of result section.

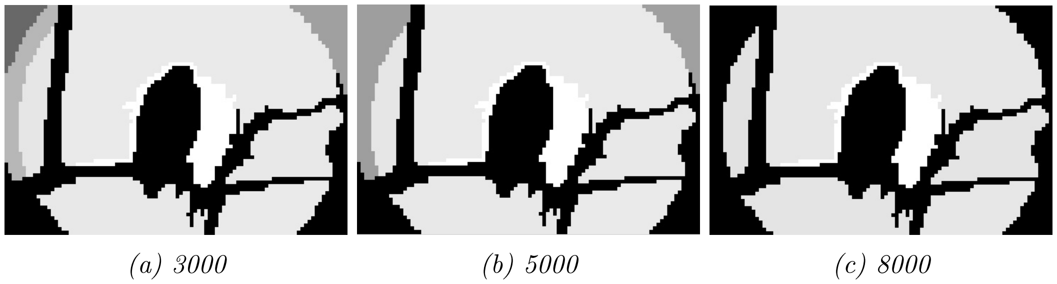
4.2 Image Model Evaluation

We evaluated the difference in performance of the merging algorithm for the i.i.d. and Markov chain image models. The merging cost in both models was varied. Figure 4.4 and Figure 4.5 shows the merged image for the i.i.d. model and Markov chain model, with various maximum merging cost.



(a) 3000 *(b) 5000* *(c) 8000*

Figure 4.4: Image 1 merged with 3000, 5000, and 8000 maximum merging cost for the i.i.d. model.



(a) 3000 *(b) 5000* *(c) 8000*

Figure 4.5: Image 1 merged with 3000, 5000, and 8000 merging cost for the Markov model.

Qualitatively, the Markov chain image model produces more meaningful segmentations as the shape of the bird is more accurately captured. In addition, the Markov model has significantly fewer remaining regions than the i.i.d. model for 3000 and 5000 maximum merging cost.

Figure 4.6 shows the PSNR vs. merging cost of the i.i.d. model to the Markov model for image 1.

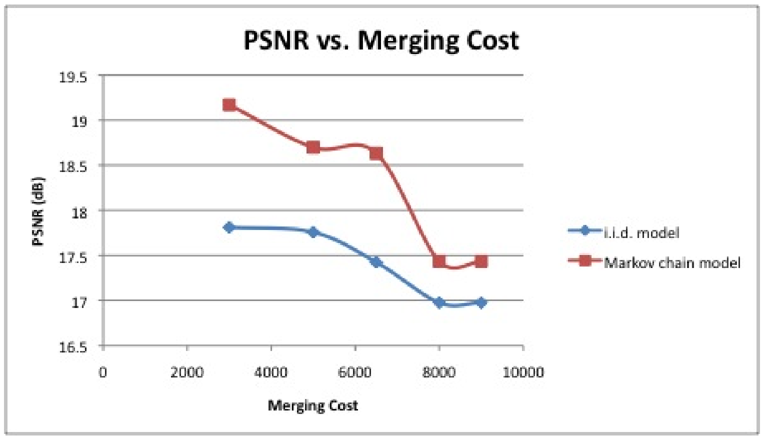


Figure 4.6: PSNR versus maximum merging cost for image 1 merged with both the i.i.d. model and the Markov model.

As it can be seen from Figure 4.6, for all maximum merging cost the Markov model has larger PSNR values and hence produces more accurate quantitative results.

Therefore, Figures 4.5 and 4.6 clearly show that the Markov model gives better qualitative and quantitative results for image 1.

4.3 Evaluation of Noisy Images

The algorithm was evaluated with salt and pepper noise and additive white Gaussian noise for both the i.i.d. model and the Markov chain model. A median filter was applied to the noisy images either one or five iterative times. The noise intensity was varied from 2% to 30% for salt and pepper noise, and from 0.02 to 0.3 variance for additive white Gaussian noise. The results for salt and pepper noise are shown using image 2 and additive white Gaussian noise are shown using image 3.

4.3.1 Salt and Pepper Noise Evaluation

i.i.d. Model

Figure 4.7 shows image 2 merged for the i.i.d. model with various noise intensities of salt and pepper noise.

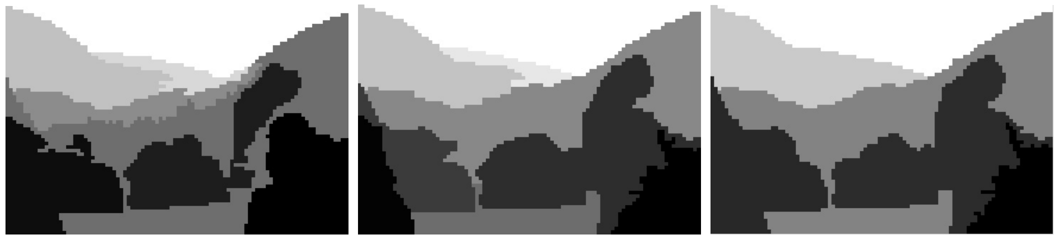

(a) 2% noise *(b) 5% noise* *(c) 20% noise*

Figure 4.7: Image 2 merged with 2%, 5%, and 30% noise intensity for salt and pepper noise for the i.i.d. model.

Figure 4.8 shows the the difference when applying none, one or five iterative median filters.

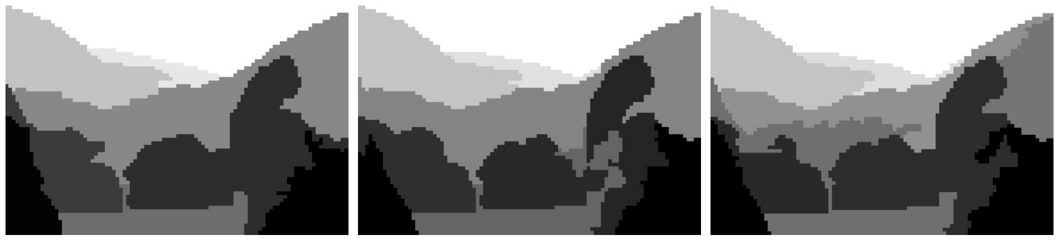

(a) Without filter *(b) One filter* *(c) Five filters*

Figure 4.8: Image 2 merged with 5% noise intensity and various number of applied median filters for salt and pepper noise for the i.i.d. model.

Figures 4.7 shows that with increasing the intensity of salt and pepper noise, the final merged image has fewer regions. In this case, the number of distinguishable hills in the background have decreased from 3 to 2, and the rocks have lost their shapes. Figure 4.8 shows that for the same merging cost of 7000, increasing the number of filters causes under-segmentation in the image. By increasing the merging cost for Figure 4.8(c), the image would be more qualitatively meaningful.

Figure 4.9 shows PSNR values versus noise intensity of merged image 2 for various number of median filters applied before merging for the i.i.d. model.

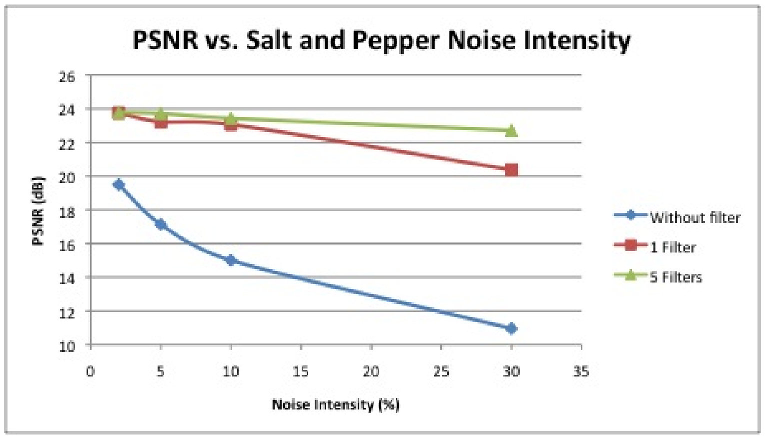
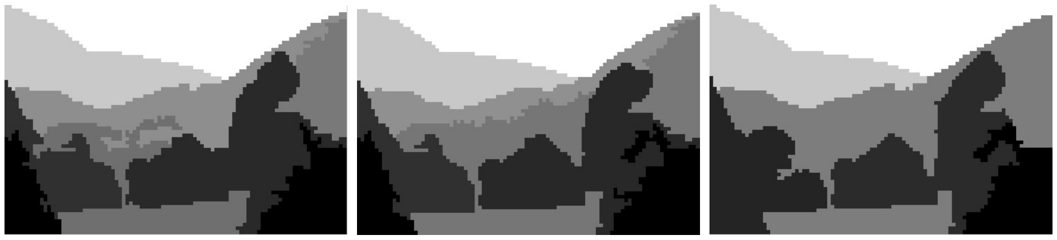


Figure 4.9: PSNR versus noise intensity for image 2 merged with teh i.i.d. model

Figure 4.9 shows that by increasing the noise intensity, the PSNR decreases and increasing the number of filters increases the PSNR. The decrease in PSNR with respect to noise intensity is more dramatic with less filters applied. We can note that an under-segmented image would result in a higher PSNR by the definition of PSNR and the method of calculating the pixel value of the merged region. Refer to section 3.6.1.

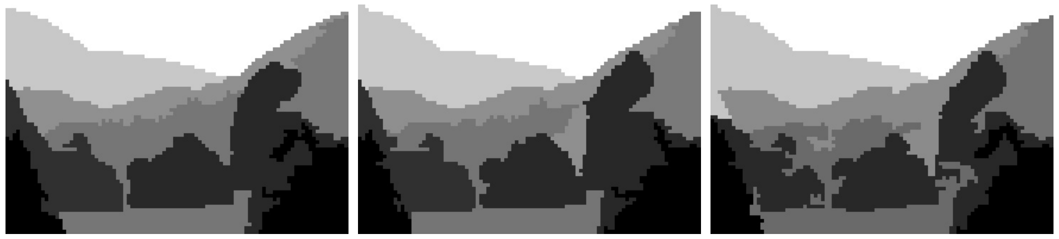
Markov Chain Model
Figure 4.10 shows image 2 with various salt and pepper noise intensities for the Markov chain model.



(a) 2% noise *(b) 5% noise* *(c) 30% noise*

Figure 4.10: Image 2 merged with various noise intensities for the Markov chain model.

Figure 4.11 shows the difference when applying none, one or five iterative median filters to the image 2.



(a) Without filter (b) One filter (c) Five filters

Figure 4.11: Image 2 merged with 5% noise intensity for various number of median filters for the Markov chain model.

In Figure 4.10, it can be seen that an increase in noise causes the algorithm to inaccurately capture the shape of the rocks. In addition, the greater the noise intensity, the less regions that remain in the merged image. This is quite similar to the i.i.d. case. Also, it can be noted that the rocks in Figure 4.7 of the i.i.d. model have a much smoother shape than those of Figure 4.10. In Figure 4.11, as the number of filters increases, the image becomes under-segmented.

Figure 4.12 shows the PSNR values for various number of applied median filters for the Markov chain model.

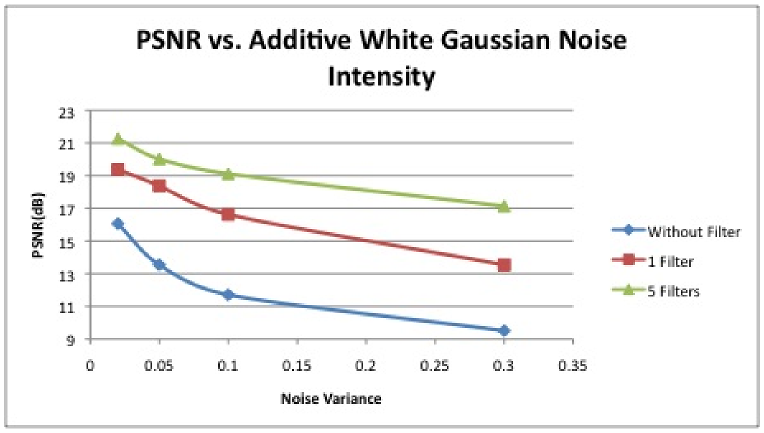


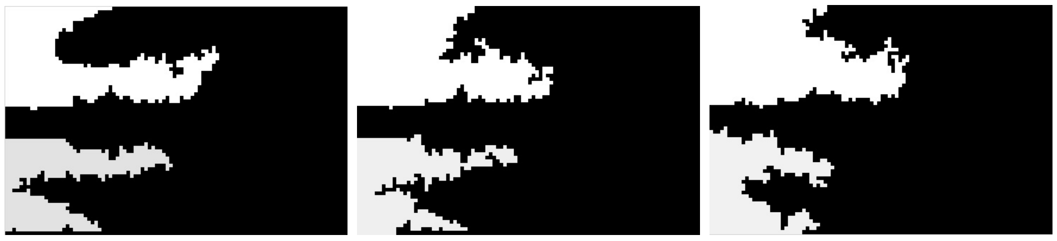
Figure 4.12: PSNR versus noise intensity for merged image 2 for the Markov model.

The under-segmentation in Figure 4.11 is reflected in Figure 4.12, where an increase in filters results in a higher PSNR value. The number of remaining regions has a direct impact on the PSNR values, as mentioned previously.

4.3.2 Additive White Gaussian Noise Evaluation

i.i.d. Model

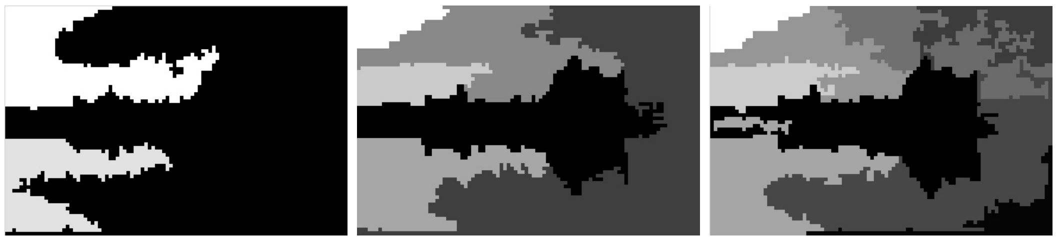
Figure 4.13 shows image 3 merged with additive white Gaussian noise for the i.i.d. model.



(a) 0.05 variance *(b) 0.10 variance* *(c) 0.30 variance*

Figure 4.13: Image 3 merged with 0.05, 0.10 and 0.30 noise variance for additive white Gaussian noise for the i.i.d. model.

Figure 4.14 shows the difference in various number of applied median filters to a 0.05 noise variance for the i.i.d. model.



(a) Without filter *(b) One filter* *(c) Five filters*

Figure 4.14: Image 3 merged with 0.05 noise intensity for various numbers of median filters for the i.i.d. model.

It should be noted that in all cases in Figure 4.13, the image is over-segmented and thus is not a meaningful representation of the original image. In Figure 4.14, as the number of filters applied increases, the merged image becomes more segmented. In Figure 4.14(c) the image is under-segmented. Thus, the merging cost was varied for a 0.05 noise variance, with five filters, to understand the results of applying multiple filters.

Figure 4.15 shows the PSNR values versus the increasing noise variance for a different number of applied median filters.

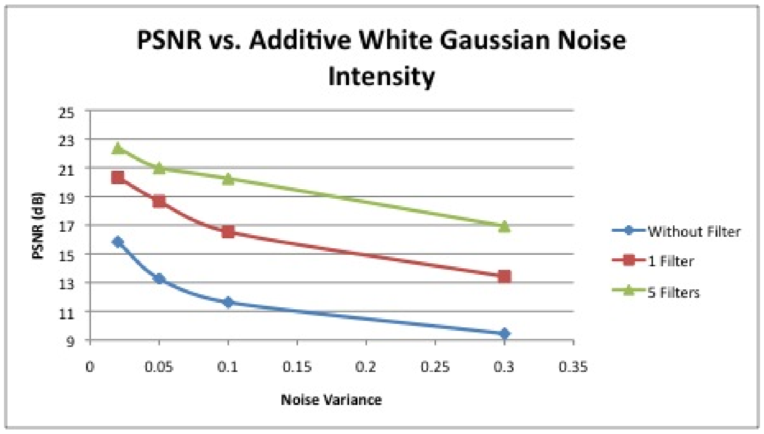
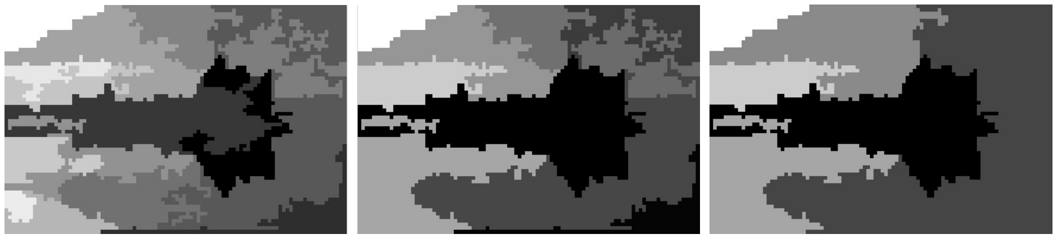


Figure 4.15: PSNR versus noise intensity for varied number of median filters for the i.i.d. model.

Figure 4.15 quantitatively confirms the qualitative assessment as mentioned previously. As the number of filters increases, the number of regions increases causing a greater PSNR value.

Figure 4.16 shows image 3 merged with a 0.05 noise variance with five median filters iteratively applied to the noisy image. The merging cost was varied.

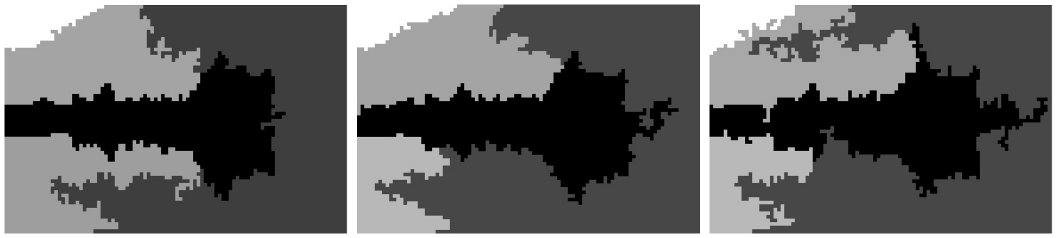


(a) 5000 *(b) 7000* *(c) 8000*

Figure 4.16: Image 2 merged with 0.05 noise variance with various merging costs for the i.i.d. model.

Markov Chain Model

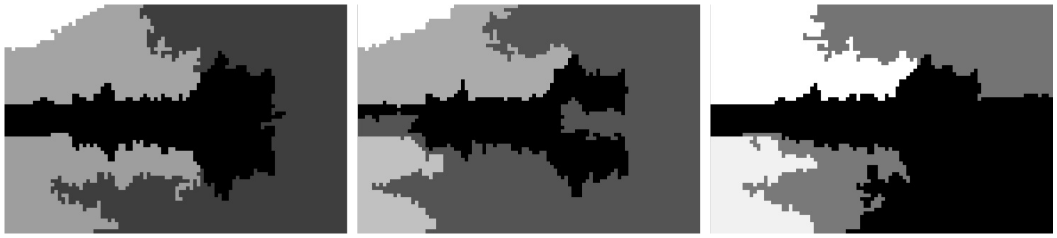
Figure 4.17 shows the results from applying different noise variance for additive white Gaussian noise without the median filter, merged using the Markov chain model.



(a) 0.05 variance (b) 0.10 variance (c) 0.30 variance

Figure 4.17: Image 3 merged with 0.05, 0.10 and 0.30 noise variance without the median filter for the Markov chain model.

Figure 4.18 shows the merged image with different number of median filters applied for 0.05 variance additive white Gaussian noise for the Markov chain model.



(a) Without filter (b) One filter (c) Five filters

Figure 4.18: Image 3 merged with 0.05 noise variance with different number of median filters for the Markov chain model.

In Figure 4.17, the number of regions within the image does not change with an increase in noise; however, the island begins to lose its shape as noise increases. There is a significant improvement in the results of the Markov chain model for merging images with additive white Gaussian noise than the i.i.d. model. In Figure 4.18, the use of the median filter does not have a positive impact on the segmentation of the image. An increase in filters causes the image to be over-segmented, differing from previous results. As discussed with the i.i.d. model, varying the merging cost could possibly remove the problem of over-segmentation.

Figure 4.19 shows the quantitative assessment of applying multiple filters to image 3 with different noise variance for the Markov chain model.

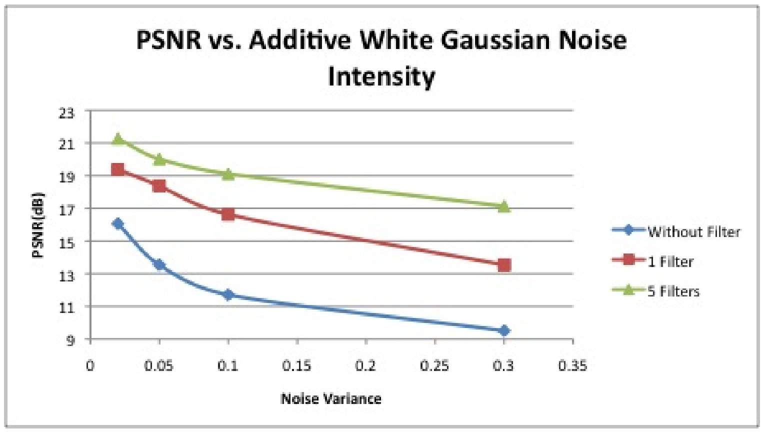


Figure 4.19: PSNR versus noise variance for different number of median filters for the Markov model with image 3.

As seen in previous results, the application of more filters results in a higher PSNR value. However, this is not indicative of better performance, as seen in Figure 4.18.

4.4 Algorithm Evaluation

As mentioned previously in section 3.5.1, the algorithms efficiency is evaluated on the amount of resources it requires for execution: time and memory.

Memory is a major constraint for our algorithm. An increase in the number of bins for quantization and a decrease in initial partition size greatly increases the amount of memory required for execution. In some cases, our algorithm requires more memory than our computers are allowed for execution, resulting in a segmentation fault. Currently, the algorithm will only run on a Macintosh and as such the amount of memory given to our program for execution cannot be increased. In order to calculate the merging cost of the Markov chain model, an additional square array with size equal to the square of the number of quantization bins of memory is needed to store the co-occurance matrix. This results in the Markov chain model requiring more memory than the i.i.d. model.

The execution time of our algorithm is negligible; however, depending on the application, the execution time could become critical. As the Markov model requires an additional array of memory, it requires more execution time. This is because the additional array must be initialized and then updated with relevant information at every merging stage. In addition, the array must be looped through for each evaluation of the merging cost. The time trials for the i.i.d. model and Markov model can be seen below for image 1 with a 10 bin quantization. For each case, both 5x5 and 10x10 initial partition size were chosen.

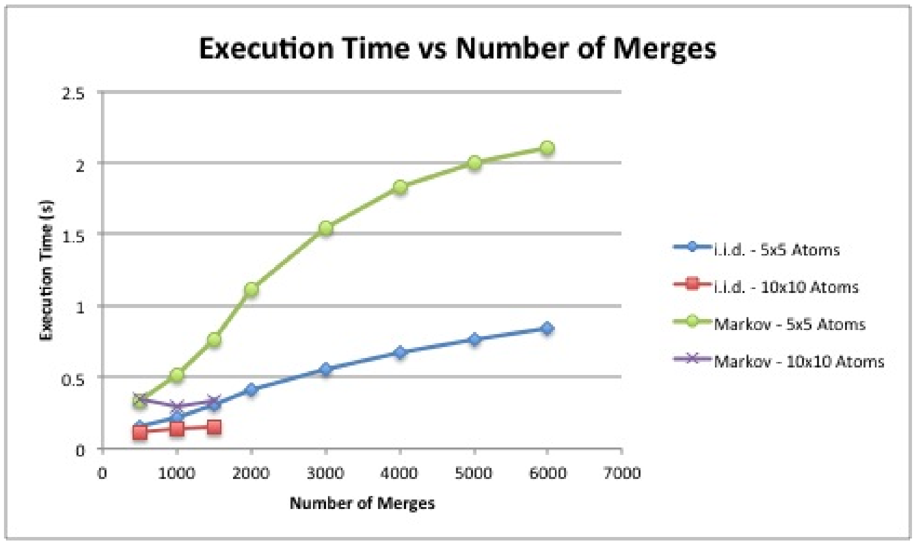


Figure 4.20: Execution time versus merging cost for different initial partition sizes and image models for image 1.

As seen in Figure 4.20, the Markov chain model requires more time than the i.i.d. model. In addition, setting a lager initial partition size decreases the execution time. This could be the result of the set-up of fewer initial partitions for the case with larger partition sizes. It is important to note that an increase in initial partition size results in less merges to obtain a segmented image.

5

Conclusion

5.1 Summary

Two stochastic processes were used to model an image: i.i.d. and Markov chain. Using both models, two types of noise, salt and pepper, and additive white Gaussian, were added to an image and the merging algorithm was tested for robustness. For noiseless images, the Markov chain model significantly outperformed the i.i.d. model both qualitatively and quantitatively, which can be seen in Figures 4.4, 4.5 and 4.6. When applying salt and pepper noise, both image models quantitatively performed better as more filters were applied to the noisy image. However, this was not reflected in the qualitative results. When applying five filters to the same noise intensity, the merged images were under-segmented. There was not a significant difference between the i.i.d. model and the Markov chain model in terms of qualitative performance for salt and pepper noisy images.

When applying additive white Gaussian noise, the i.i.d. model over-segments all tested variances of noise for the merged image. The Markov chain model significantly performed better with the same tested variances used in the i.i.d. model. Both image models quantitatively performed better as more filters were applied to the noisy image. However, the better performance was only reflected qualitatively for the i.i.d. model. The Markov chain model resulted in worse qualitative results when multiple filters were applied to an additive white Gaussian noisy image. In conclusion, the Markov model is better for the case of additive white Gaussian noise without filters, and with filters the i.i.d. model is better.

5.2 Future Work

There are a few directions that future work could take this project. One could model the image using different stochastic processes, further investigate noise, or do more extensive testing on the current algorithm.

The paper by Calderero and Marques [5], and "Design of Region Merging Algorithms for Image Segmentation" [6], discussed modelling an image as a Markov random field. Results from both papers showed that the Markov random field produced more meaningful segmented images. The Markov random field model could also be tested for results with noisy images.

There are many directions one could further investigate noisy images. With the current algorithm, the median filter size window size or shape could be adjusted. Different types of noise could be investigated, such as burst noise or damaged images. As well, one could test different types of filters for different types of noise and determine the best filter for possible noise combination.

Finally, more extensive testing could be done on the current algorithm. One could test different types of images, such as landscapes, animals, or faces. One could also test the algorithm for images with texture, intensity or colour. These images could be used to test the robustness of the algorithms. One could use different partition shapes, such as the watershed method mentioned in Calderero and Marques paper [5]. Finally, instead of removing noise before applying the merging algorithm, one could incorporate the removal of noise in each merging step.

6

Team Members

Two faculty advisors and three students and worked on this project as a team. The faculty advisors are Fady Alajaji and Abdol-Reza Mansouri; both professors at Queen's University.

Darcie Dillon
Darcie is in her fifth and final year studying Mathematics and Engineering in the Mechanics option. She did an internship between her third and fourth year at Queen's University and is travelling eastern Canada during the summer. She will find a job in mechanical engineering after graduation.

Katharine Kerr
Katharine is a fifth year student in Mathematics and Engineering in the Communications and Computing option. Three years ago, she discovered her love of software and will be pursuing a career within the industry. She is an avid outdoors enthusiast and will be exploring Europe after she graduates.

Sungbok Lee
Sung is a fifth year student in Mathematics and Engineering in the Communications and Computing option. After the completion of his internship, he is plannings to go back to the same company to work for a year or two but will eventually apply for a graduate school to study further in Robotics field, especially in motion planning area.

7

Bibliography

[1] A. Fisher, “Inside Google’s Quest To Popularize Self-Driving Cars”, Sept. 2013. http://www.popsci.com/cars/article/2013-09/google-self-driving-car

[2] “Broadband Patterned Wafer Defect Inspection Systems”, *KLA Tencor*. http://www.kla-tencor.com/front-end-defect-inspection/29xx-series.html

[3] “Apple Support ”, http://support.apple.com/kb/TA24990.

[4] “Motion Metrics ”, http://www.motionmetrics.com.

[5] F. Calderero and F. Marques, “Region Merging Techniques Using Information Theory Statistical Measures”, IEEE Trans. Image Process., vol. 19, pp. 1567 1586, Jun. 2010.

[6] R. Cannon, A. Lappalainen, and R. Zurkowski, “Design of Region Merging Algorithms for Image Segmentation” 4th year project, Mathematics and Engineering, Queen’s University, 2012.

[7] D. Blair, V. Buckingham, and A. Goldberg, “Design of Region Merging Algorithms for Image Segmentation with an Extension to Splitting”, 4th year project, Mathematics and Engineering, Queen’s University, 2013

[8] “medfilt2”, *Mathworks*
http://www.mathworks.com/help/images/ref/medfilt2.html

[9]“Removing Noise From Images”, *Mathworks*,
http://www.mathworks.com/help/images/removing-noise-from-images.html

Appendix A

Images

Three images were used to test the robustness of the algorithm, as well as study noisy images. All images were quantized with a 10 bin quantizer, and where applicable a 5x5 initial partition size. The three images tested were:

Image 1: Bird on a branch

Image 2: Rocks with hills in the background

Image 3: Island on a body of water

Image 1 was tested using a merging cost of 5000, and images 2 and 3 were tested using a merging cost of 7000.

The C and MATLAB code for the algorithm can be found on a CD. The following images and graphs are results from testing the algorithm.

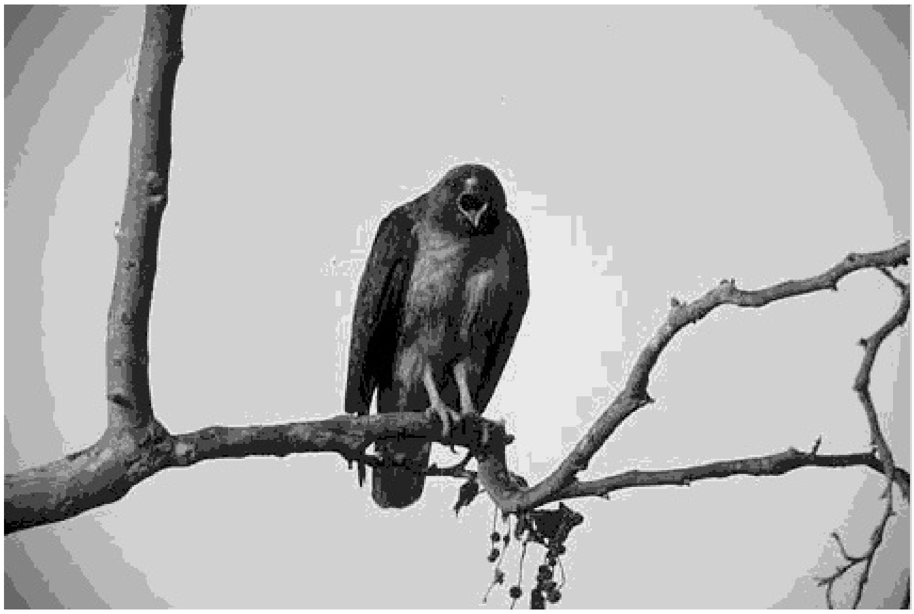


Figure A.1: Image 1 with a 10 bin quantizer.

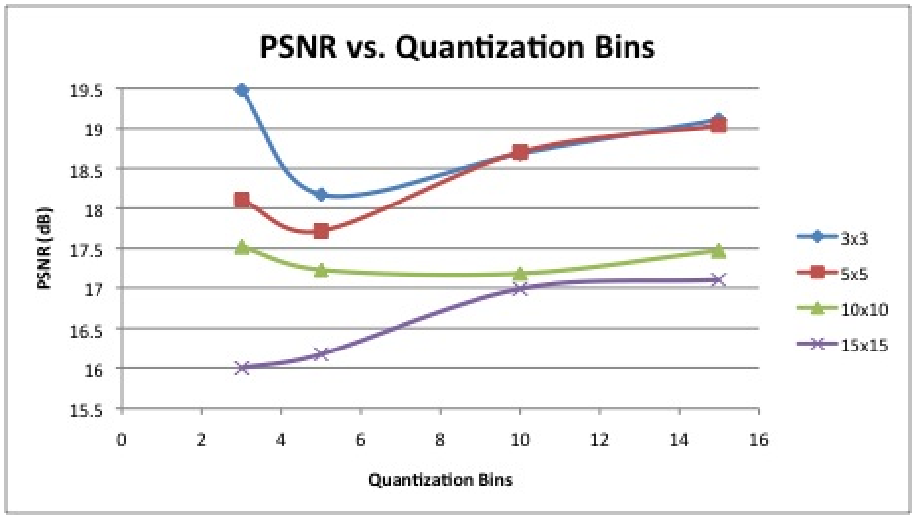


Figure A.2: PSNR versus quantization bins for different sized initial partitions for merged image 1.

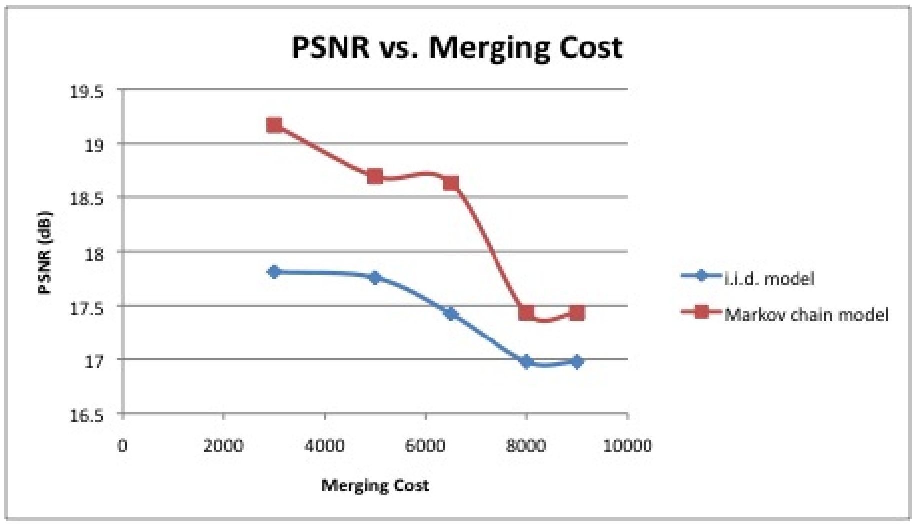


Figure A.3: PSNR versus merging cost for i.i.d. model and Markov model for image 1.

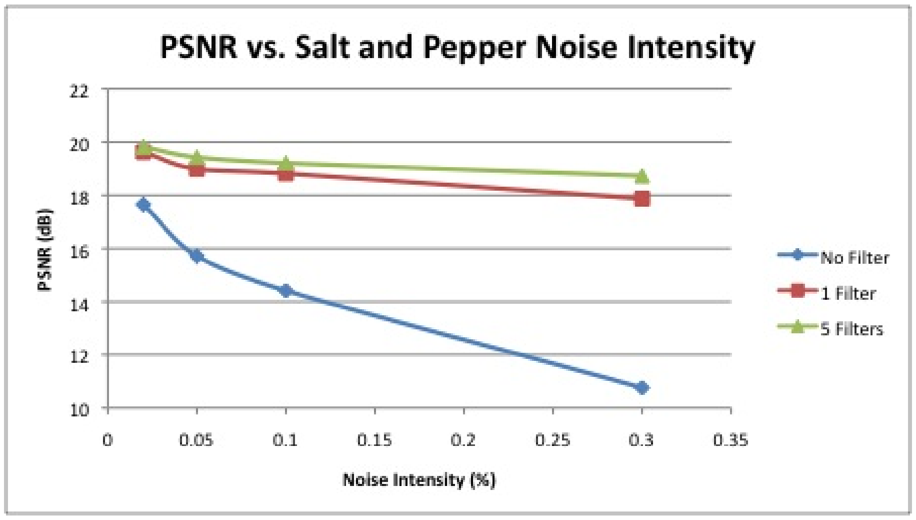


Figure A.4: PSNR versus noise intensity for salt and pepper noise for zero, one or 5 iterative median filters applied for image 1.

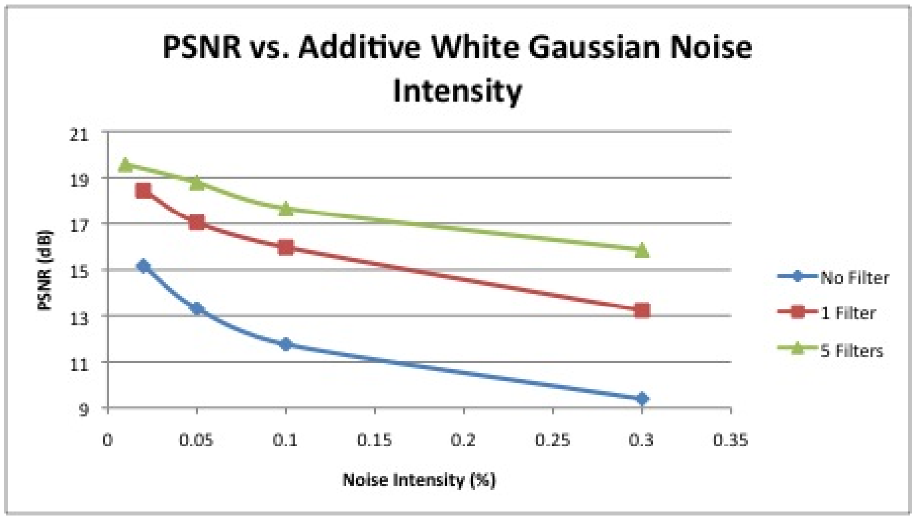


Figure A.5: PSNR versus noise intensity for additive white Gaussian noise forzero, one or 5 iterative median filters applied for image 1.

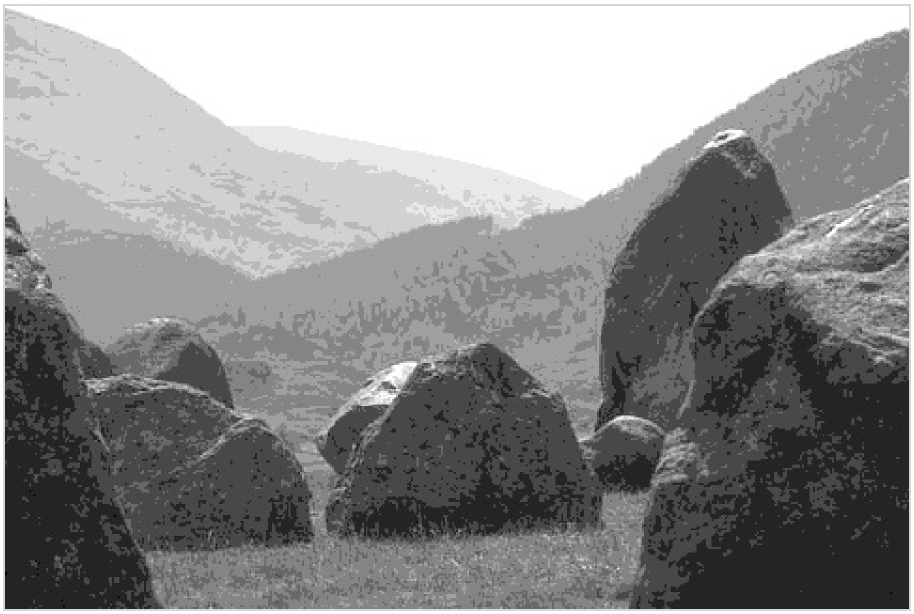


Figure A.6: Image 2 with 10 bin quantizer.

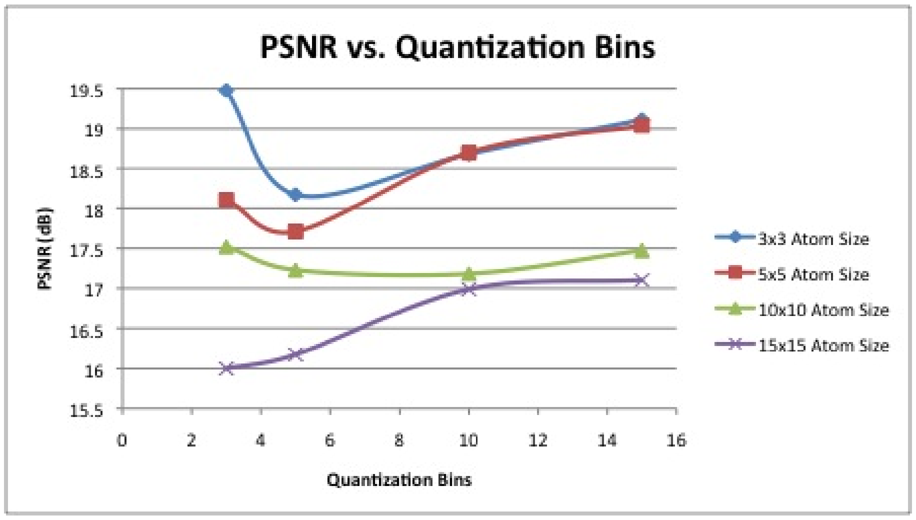


Figure A.7: PSNR versus quantization bins for different sized initial partitions for merged image 2.

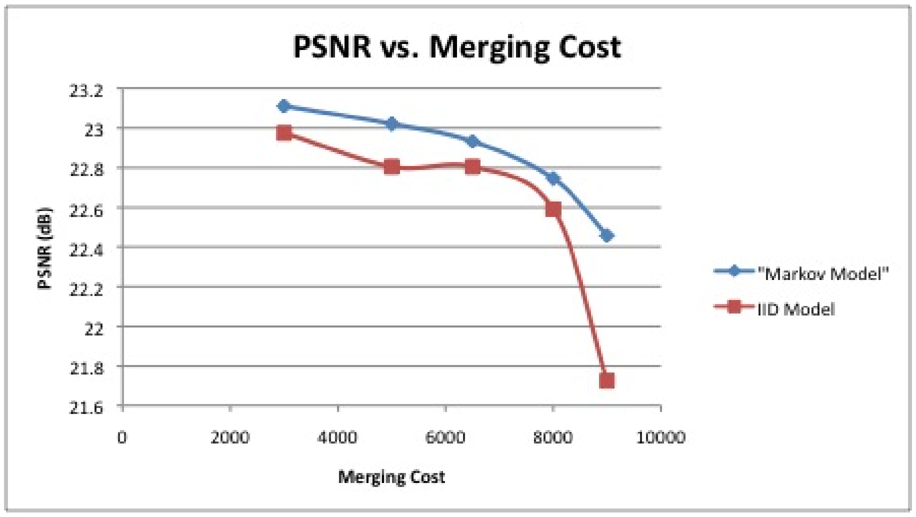


Figure A.8: PSNR versus merging cost for i.i.d. model and Markov model for merged image 2.

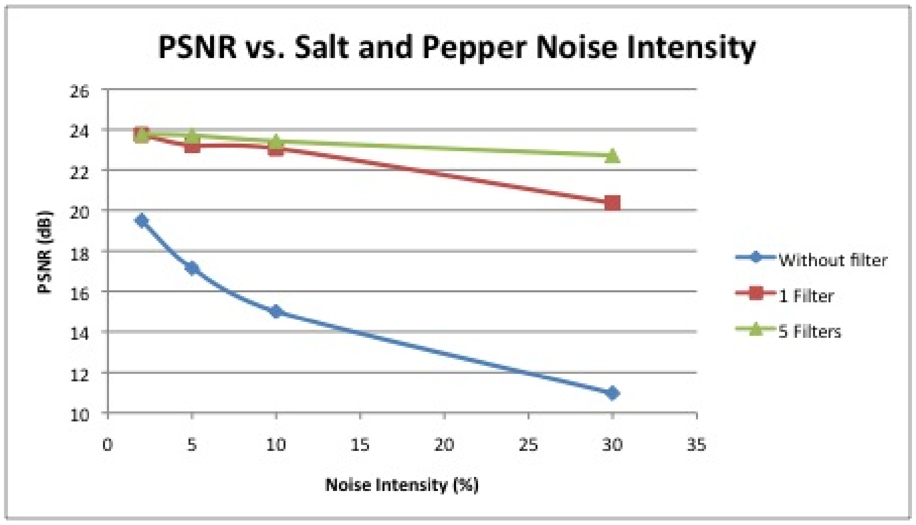


Figure A.9: PSNR versus noise intensity for salt and pepper noise for zero, one or 5 iterative median filters applied for image 2.

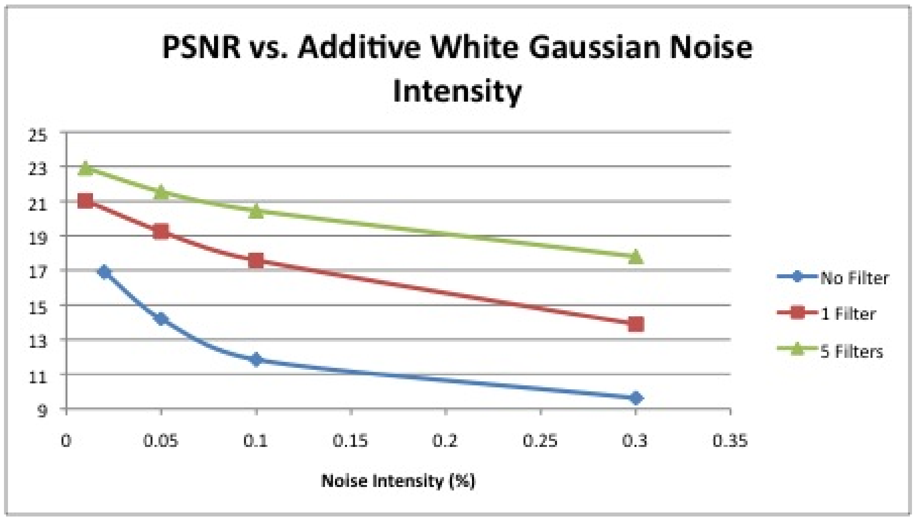


Figure A.10: PSNR versus noise intensity for additive white Gaussian noise for zero, one or 5 iterative median filters applied for image 2.

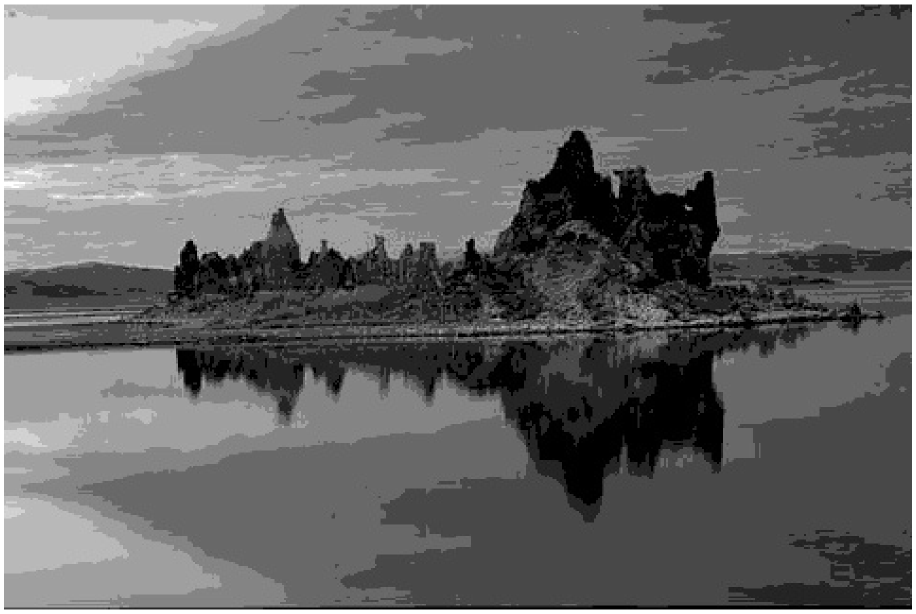


Figure A.11: Image 3 with 10 bin quantizer.

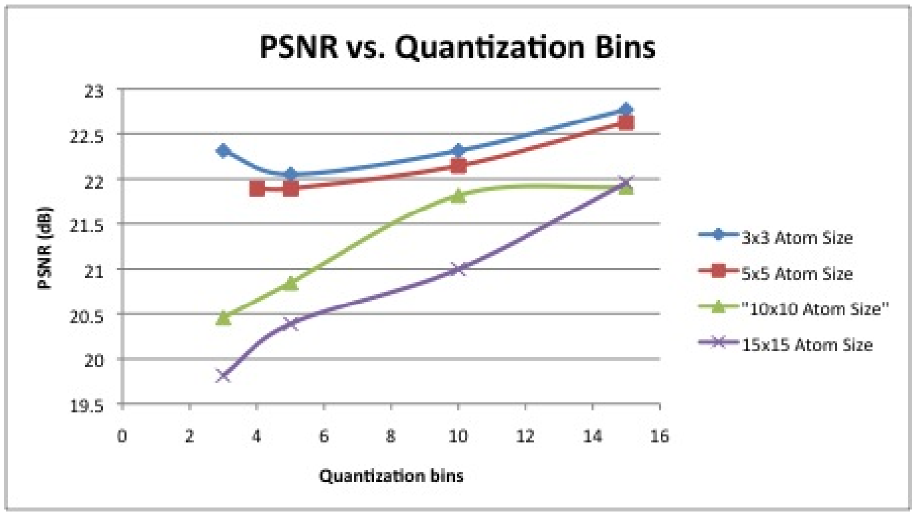


Figure A.12: PSNR versus quantization bins for different sized initial partitions for image 3.

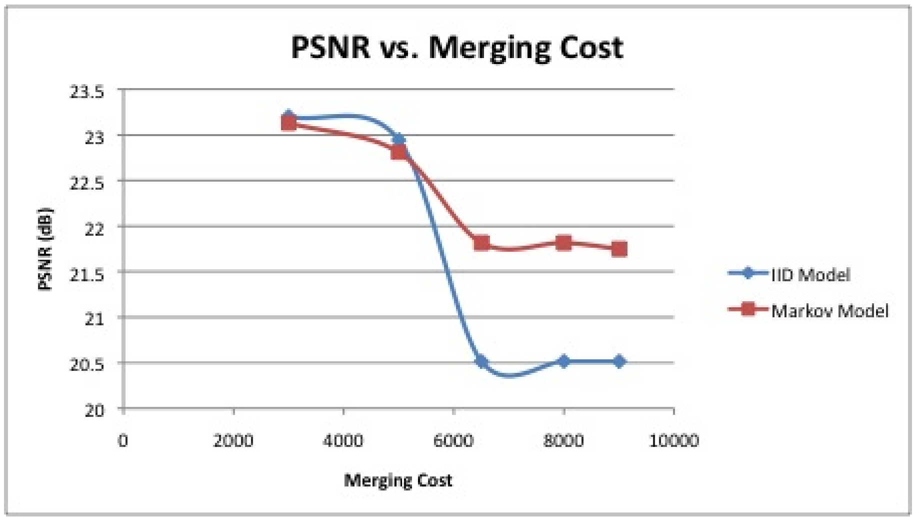


Figure A.13: PSNR versus merging cost for i.i.d. model and Markov model for merged image 3.

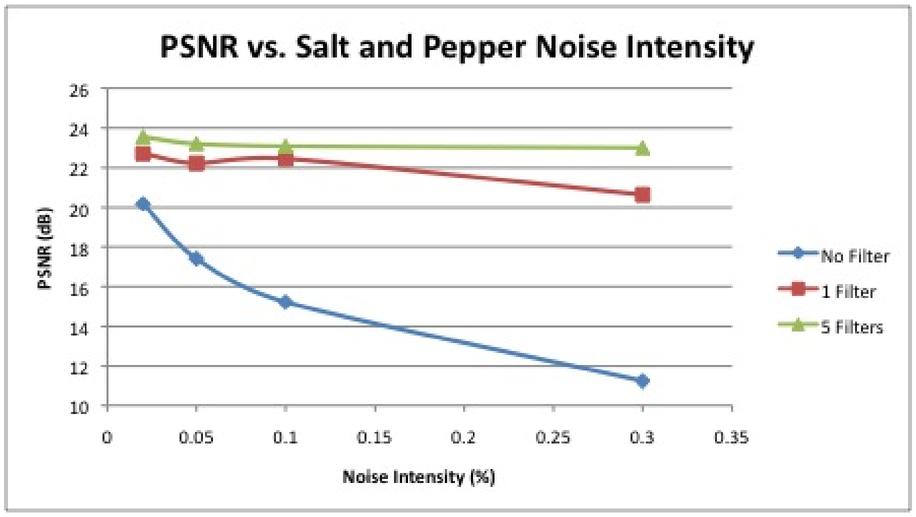


Figure A.14: PSNR versus noise intensity for salt and pepper noise for zero, one or 5 iterative median filters applied for image 3.

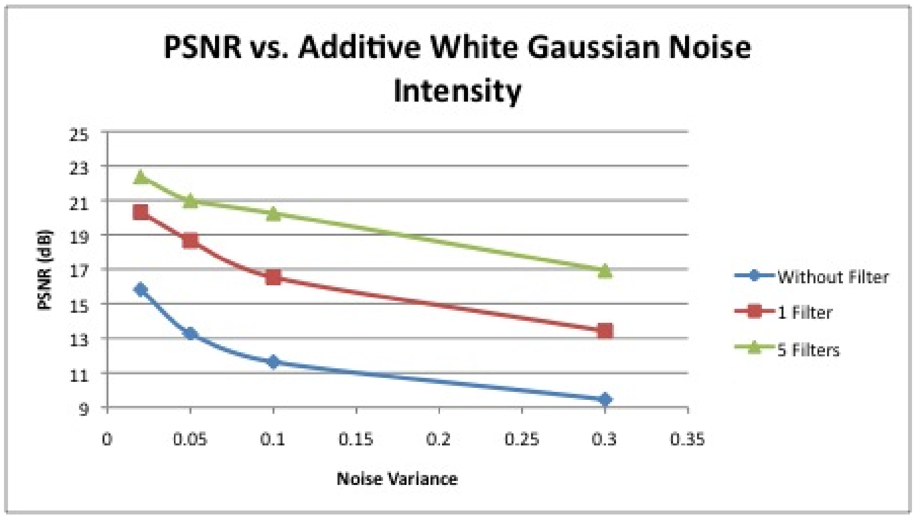


Figure A.15: PSNR versus noise intensity for additive white Gaussian noise for zero, one or 5 iterative median filters applied for image 3.