Grayscale Image Colorization Using Machine Learning Techniques

Zachary Frenette ZFRENETT@UWATERLOO.CA
University of Waterloo, 200 University Ave W, N2L 3G1

Abstract

From modern medical imaging to antique photography, there exists vast amounts of illustrations and photographs that lack color information. Adding color to these images could help improve both visual appeal and expressiveness. Given that the majority of colorization methods rely heavily on user interaction, we would like to explore how machine learning techniques can be applied to the colorization process of grayscale images, and then analyze the limitations of each of these techniques.

1. Introduction

Image colorization can be described as the process of assigning colors to the pixels of a grayscale image. This problem is ill-posed in the sense that, without prior information regarding the image, there is often more than one possible colorization. In other words, the colors of an object cannot usually be distinguished from one another by simply looking at the grayscale component of the object. For example, all other things equal, a red balloon would most likely look the same as a green balloon in a grayscale image. Because of this property, automatic image colorization is a very challenging task.

Many of the current methods used for grayscale image colorization rely heavily on user interaction. Normally, this is achieved by professional artists who use software to manually adjust the colors, brightness, contrast and exposure of the image (Pitié et al., 2008). Not only is this an expensive procedure, but it is also very time consuming. On the other hand, the semi-automatic algorithms that exist for image colorization all suffer from a variety of limitations. These limitations range from a lack of robustness, to a requirement for some of the data to be manually processed. For example, there has been some work done for the case where the user provides the colors of a few regions before having an algorithm propagate this color information to the rest of the image (Levin et al., 2004). Similar work has been done for which the user plays more of an interactive role by manually coloring some of these regions in between propagation steps (Charpiat et al., 2010). There has also been some work done in the context of fully-automatic algorithms, though the algorithms proposed only seem to work well when the image has a few colors (Ashikhmin et al., 2002). In addition, several discrepancies can typically be observed in the resulting images.

Recently, machine learning techniques have been employed in the colorization process of grayscale images (Charpiat et al., 2010; Liu and Zhang, 2012). In this project, we compare the performance of some these machine learning methods, and then analyze their respective limitations. Section 2 discusses some of the models that are required in order to present the colorization methods that we consider. In sections 3 and 4, we formulate image colorization as a machine learning problem and describe the dataset that will be used throughout this project. In sections 5 and 6, we provide a discussion of the methods employed as well as the results obtained, while in section 7, we finish with some concluding remarks and potential directions for future work.

2. Preliminaries

In this section, we define several of the models that will be used in the development of our image colorization algorithms.

2.1. The Spatial Image Model

There are several different ways of thinking about the representation of an image. Intuitively, we can think of an image as a function $f : U \to \mathcal{C}$ where $U \subset \mathbb{R}^2$ is a subset of the plane and $\mathcal{C}$ is a color space. However, from a computational perspective, this may not be the

Project report for CS886: Applied Machine Learning. University of Waterloo, Fall 2014.

ideal representation of an image. Instead of trying to encode a continuous subset of $\mathbb{R}^2$, we will encode a discrete subset U' of U. More specifically, we will consider a natural discretization in the spatial image model (Velho et al., 2008). In this model, we assume that the domain of f is a rectangle $U = [a, b] \times [c, d]$, and for some fixed values of δ_x and δ_y, we apply the following discretization:

$$U' = \{(x_j, y_k) \in U : x_j = j\delta_x,\ y_k = k\delta_y \text{ where } j, k \in \mathbb{Z}\}.$$

Here, U' is an orthogonal lattice of points, where each point (x_j, y_k) is called a pixel. Using this definition of U' yields a natural matrix representation of the image. More specifically, we can define a matrix $I \in \mathcal{C}^{m \times n}$ where each element $I_{j,k} = f(x_j, y_k)$. In other words, the value of each element in our matrix is simply equal to the color of that corresponding pixel. Given that images on a computer are typically encoded using this type of representation, we will adopt this representation throughout the remainder of this project.

2.2. The LAB Color Model

Images on a computer are typically represented in the RGB color model, in which the color of each pixel is determined by a 3-tuple $\rho = (r, g, b)$ denoting the red, green, and blue components of that color respectively. However, the RGB model is a device dependent color model, and therefore, some of the color points that are represented in this space may not be absolute. Hence, the same point may produce different colors on different devices. Furthermore, this color model was optimized for performing operations on devices and therefore is not representative of human perception (Velho et al., 2008). Because of these properties, we will instead choose to represent images in the LAB color model (Charpiat et al., 2010). The colors in this model comprise of 3 different components, the first of which represents luminance while the other two orthogonal components store explicit color information.

Not only is the LAB color model device independent, but it was also designed in a way to approximate the human perception of brightness and color. In particular, the Euclidean distance between two colors in this model approximates the difference in perceived color, which provides a natural distance metric for measuring the similarity between two colors. It is worth noting that the color gamut of this model is larger than the color gamut of human vision. This means that there are points in this space that do not correspond to any of the colors that are perceivable by the human eye. For additional details regarding the different characteristics of these two color models, we refer the interested reader to the work of Hunt (Hunt, 2005).

2.3. Markov Random Fields

A Markov random field is a graphical model which has the structure of an undirected graph. The vertices of this graph correspond to random variables while the edges model the conditional independencies between them. More formally, a Markov random field is a tuple $\mathcal{M} = (V, \mathcal{F}, \Lambda, N)$, where $V = \{v_1, \ldots, v_n\}$ is a set of vertices, $\mathcal{F} = \{F_i : i \in V\}$ is a set of random variables, Λ is a set of labels, and $N : V \to 2^V$ is a neighborhood function (Bishop, 2006). The set Λ contains a label for each possible outcome of the random variables contained in $\mathcal{F}$. In particular, we will use the notation $\lambda_i \in \Lambda$ to denote the label assigned to the random variable F_i. Furthermore, we will use the notation $F_S = \{F_i : i \in S\}$ and $\Lambda_S = \{\lambda_i : i \in S\}$ to denote specific sets of random variables and labels respectively.

A Markov random field must also satisfy the Markov property. That is, for any particular realization of the random variables, we have that:

$$P(F_i = \lambda_i | F_{V-i} = \Lambda_{V-i}) = P(F_i = \lambda_i | F_{N(i)} = \Lambda_{N(i)}).$$

In other words, F_i is independent of any other random variable given its neighbors. Markov random fields are often used to model labeling problems where some particular labeling is desired. In our case, we will be interested in labeling the pixels of a grayscale image with colors from our color space $\mathcal{C}$. From an algorithmic point of view, this desired labeling is obtained by trying to find an assignment of colors that minimizes an appropriate energy function $\mathcal{E}(\Lambda_V)$. Markov random fields have many other useful properties, though not all of them will be needed for this project (Wang et al., 2013). The properties that will be needed will be discussed in section 5 when the details of the machine learning algorithms are presented.

3. Machine Learning Formulation

There are several differences between the traditional supervised learning paradigm and the one we will adopt for this project. Under typical circumstances, we are given a fixed training set where the training examples are sampled independently from some underlying distribution $\mathcal{D}$. This set is then processed by some learning algorithm before it can be used to classify new data points, which are also sampled independently from this same distribution. In our

case, we are given a collection of colored images $\mathcal{I} = \{I_1, \ldots, I_k\}$ as well as a grayscale image I', and the goal is to assign a color from $\mathcal{C}$ to each of the pixels of I'. It is worth observing that $\mathcal{I}$ plays the role of our training set while the pixels of I' represent the new data points that we wish to label and classify.

Unlike in the traditional supervised learning paradigm, our training set $\mathcal{I}$ is of small size and its content varies depending on the grayscale image I'. More specifically, our training set is chosen to be a small set of colored images which are all similar to the grayscale image that we are trying to colorize. For our project, the training set will consist of a single colored image, though it is possible to consider cases where the training set contains several images. As a result, for each of the grayscale images that we want to colorize, a new training set needs to be chosen and the learning algorithm needs to be re-executed.

4. Dataset Used

Our dataset will primarily consist of a subset of the colored images made available by Jégou, Douze and Schmid from their work on image retrieval (Jégou et al., 2008). More specifically, this dataset entails a wide variety of outdoor images, some of which include natural scenery, man-made objects, animals, lakes and waterfalls. Furthermore, this collection contains images that have been rotated, and that contain changes in perspective or illumination. These changes will help us test the robustness and limitations of the machine learning algorithms we apply. Lastly, it is worth noting that, although primarily outdoor scenes, this dataset contains a handful of indoor images as well.

For testing purposes, we will select a small subset of the images and transform them into their grayscale counterpart. Although in practice we will not know the real colorings of these images, it will facilitate the task of measuring the error and performance of each machine learning algorithm. Reasonable error metrics for the case in which the true colorization of an image is not known still requires additional research.

5. Description of Methods Used

In this section, we describe each of the methods and steps used in our image colorization algorithms.

5.1. Preprocessing Step

In order to extract a meaningful set of features and facilitate training, each image in our training set will be preprocessed in two ways. As described in section 2.2, we begin by converting the representation of each image from the RGB color model to the LAB color model. This conversion is accomplished through a series of non-linear transformations, although the details of said transformations are not important for this project (Hunt, 2005).

Next, we reduce the size of the color space of each image to a manageable subset of colors through a process called color space quantization (Charpiat et al., 2010; Velho et al., 2008). A typical image in our dataset has tens of thousands of different colors, many of which only appear in a small handful of pixels. Hence reducing the size of our color space will not only help eliminate outliers, but also allow us to work with significantly less prediction classes. Using an algorithm like k-means, we can cluster groups of contiguous pixels having similar colors into k different bins (Bishop, 2006). By assigning a color to each of the bins, we can then recolor the image by only using k different colors. It is worth noting that we are only performing quantization on the (a, b)-components of each pixel since the luminance component does not store explicit color information. For this project, we will consider $k = 16$ different color classes.

5.2. Feature Extraction

In essence, we are interested in labeling pixels with their appropriate color. Hence the features that we select should reflect the properties of the pixels instead of the entire image. However, features on individual pixels do not convey much information. Instead, for a given pixel ρ of the image, we will extract features from a $\delta \times \delta$ window centered at ρ. This will allow us to obtain information about the local neighborhood of ρ. As a result, we will be interested in 4 classes of features over this $\delta \times \delta$ window: SURF descriptors (Bay et al., 2006), the magnitude of the 2D Discrete Fourier Transform (DFT), the grayscale histogram, and the localized mean and standard diviation of the intensity. In particular, we will calculate SURF descriptors over three different scales, while the other features are extracted over a 11×11 window centered at ρ. This gives us an ungodly 763 dimensional feature vector.

As the dimensionality of our feature vector is quite large, we will apply the Principal Component Analysis (PCA) algorithm in order to reduce its dimensionality (Bishop, 2006). In particular, we will reduce the dimensionality in a way such that 90% of the variance in our data is maintained. For each image, feature vectors will be extracted from a random sample of N pixels, which we choose to be approximately 4% of the

pixels in the image.

5.3. Initial Color Prediction Phase

Predicting the colors of the pixels in our grayscale image is done in two phases. In the first phase, we obtain initial estimates for the probabilities of the colors of each pixel. After obtaining these estimates, the image is modeled as a Markov random field where graph cuts are used in order to obtain a globally spatial coherent labeling of the pixels. This section will address the first phase of the colorization process while section 5.4 will discuss the application of graph cuts in obtaining a final coloring of the image.

In order to estimate the desired probabilities, we consider two different machine learning models. The first model that we consider is linear logistic regression. Let $\mathcal{C}' = \{c_1, c_2, \ldots, c_k\}$ denote the quantized color space after applying k-means clustering and let ϕ_ρ denote the feature vector for pixel ρ. Under this model, we are interested in learning linear decision boundaries that model $P(C = c_i|\phi_\rho)$ where c_i is a color in $\mathcal{C}'$. Although simple, linear logistic regression can be parameterized to control regularization and thus should provide a good baseline. It is worth noting that the implementation of linear logistic regression selected is using a one versus all approach rather than a true multinomial regression technique. In particular, k different logistic regression models are trained as follows. For each model, the data is divided in a way such that all feature vectors with output class c_i are grouped together as positive instances while the other feature vectors are treated as negative instances. However, this approach creates a skew between positive and negative instances. Therefore, in order to help minimize bias, examples are sampled inversely proportional to their frequencies in the training set.

The second model we consider is support vector machines, which are one of the most popular classifiers used in the literature for image colorization (Charpiat et al., 2010). In this model, we are interested in learning the decision boundaries directly rather than first learning the conditional probability distribution $P(C = c_i|\phi_\rho)$. Once again, a one versus all approach is taken for classification. That is, we train k support vector machines that perform binary classification, and the training data is divided in a manner that is analogous to what is described for linear logistic regression. To allow for more flexibility, a Gaussian kernel is used to create non-linear boundaries. In addition, as images are noisy by nature, soft-margin classifiers are used in order to allow a small degree of misclassifications. Since we are not learning the distribution $P(C = c_i|\phi_\rho)$ directly, we will use the distance between a new data point and the margin as a proxy for confidence. These values are then used in the post-processing phase by the graph cut algorithm.

It is worth mentioning that there are other simple machine learning models that can be used to perform multiclass classification, such as nearest neighbors and decision trees (Bishop, 2006). However, the main reason they were not considered as models for image colorization is that there is no straightforward way to obtain probability estimates for the different color classes. We will return to this idea when we discuss possible avenues for future work.

5.4. Post Processing Phase Using Graph Cuts

The colorization process of a grayscale image I' starts by computing the feature vectors ϕ_ρ for every pixel in I'. These feature vectors are then passed to one of the models described in the previous section where probability estimates are derived for each color class. In order to achieve a globally spatial coherent coloring, the grayscale image is then modeled as a Markov random field $\mathcal{M} = (V, \mathcal{F}, \Lambda, N)$. In particular, we begin by setting a vertex and a random variable for every pixel in our grayscale image. That is, we define the sets $V = \{v_\rho : \rho \in I'\}$ and $\mathcal{F} = \{F_{v_\rho} : v_\rho \in V\}$. The labels of our Markov random field correspond to the possible colors that a pixel can have. In our case, we have that $\Lambda = \{\lambda_c : c \in \mathcal{C}'\}$. The last thing to define is our neighborhood function, which is what models the edges in our graph. When building a Markov random field, we assume that our graph satisfies the Markov property. As such, there are several reasonable choices for a neighborhood function when modeling an image (Boykov and Veksler, 2006). For our application, we will choose a neighborhood function such that every vertex is connected to the 8 other vertices surrounding it.

In essence, a graph cut algorithm works by trying to find an assignment of labels that minimizes some energy function defined on our Markov random field (Boykov and Veksler, 2006). In our case, this energy function should accomplish two things. First, we would like the coloring between adjacent pixels to be smooth, while allowing color discontinuities at edge boundaries. Second, we would like to encourage a labeling of pixels by colors that were initially predicted with high confidence. Let $\Lambda_V = \{\lambda_\rho : v_\rho \in V\}$ denote some labeling of the vertices and let $g(\rho)$ denote the magnitude of the gradient at pixel ρ. In addition, we define $s(\rho, \lambda)$ to be the estimated probability that ρ has

the color labeled by λ. With these auxiliary terms, we can define our energy function $\mathcal{E}(\Lambda_V)$ as follows:

$$\mathcal{E}_\rho(\Lambda_V) = \sum_{\rho'} g(\rho') \cdot \|\lambda_\rho - \lambda_{\rho'}\|_2 - \sum_{\lambda \in \Lambda} s(\rho, \lambda)\delta(\lambda_\rho, \lambda)$$

$$\mathcal{E}(\Lambda_V) = \sum_{\rho} \mathcal{E}_\rho(\Lambda_V).$$

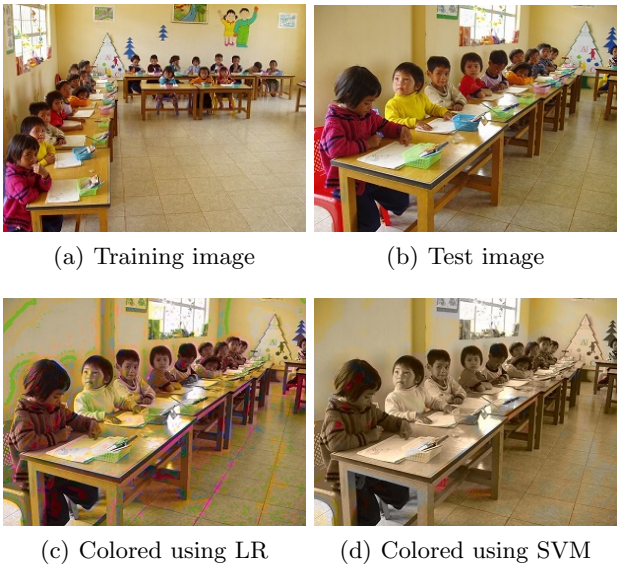
Here, ρ' is used to denote the neighbors of ρ according to our neighborhood function N, and δ is a function that produces 1 if and only if its two parameters are equal. The first term in our summation is used to penalize color variation where it is not expected, while the second term encourages the use of high confidence colors.

In our project, we have made use of the graph cut implementation provided by Delong, Osokin, Isack, and Boykov (Delong et al., 2012). Despite the fact that finding an optimal solution is NP-Hard, their algorithm can approximate the solution within a reasonable amount of time.

6. Error Analysis and Discussion

Our experiments have produced mixed results. In particular, the quality of the coloring obtained varies greatly from image to image and depends heavily on the choice of parameters. For images containing generic outdoor scenery, the algorithms tend to do reasonably well. Although several patches are colored incorrectly, the general colors are all present. However, for most indoor images with many different colors, the two algorithms perform quite badly. In particular, Figure 1 shows the results of a colorization that both models struggled with. We can observe that several important colors are missing in the results, as well as heavy patches of noise and discoloration. One possible explanation for these difficulties is that the image has many colors that are only present in small contiguous areas. Therefore, since we are sampling our training examples randomly, it could very well be the case that we are not extracting enough information from those regions. This is made evident by noticing that, especially for the support vector machine model, most of the image is colored yellow, the dominant color found in our training image. These observations suggest that a better sampling method may be required in order to achieve better results.

Another common source of error that we observe is the inability to differentiate between regions that have similar textures but different colors. Since our features are all extracted within a small window surrounding ρ, we do not capture differences between two regions that are locally similar, but that are part of differently colored objects. For example, we can see that in Figure 2, both our models had difficulties differentiating between the ragged green leaves and some of the mountainous regions. In fact, these difficulties are much more apparent for the logistic regression model than they are for the support vector machine model. This suggests that simply incorporating localized features for each pixel may not be enough to achieve good colorizations. For example, one might try explicitly incorporating an object recognition step or a region segmentation step. This would allow us to add features that keep track of which region or which object each pixel belongs with.



(a) Training image (b) Test image

(c) Colored using LR (d) Colored using SVM

Figure 1. Sample coloring of a school classroom

Another important observation is that our algorithms appear to be robust against certain transformations. In particular, we have observed that both models seem to perform well when the test image is more or less a rotation of the training image. This is somewhat expected because a large portion of our features are obtained from SURF descriptors, which were originally designed to be unaffected by rotations (Bay et al., 2006). One such example can be seen in Figure 3. Besides for a few discolored patches, both models deliver promising results. On the other hand, our algorithms perform quite poorly when there is a significant difference between the brightness of the training image and the test image. A change in brightness affects the luminance component of every pixel, and hence changes a large portion of our feature vector. The Fourier transform, the grayscale histogram, and the mean of the intensity

(a) Training image (b) Test image

(c) Colored using LR (d) Colored using SVM

Figure 2. Sample coloring of a mountainous region

(a) Training image (b) Test image

(c) Colored using LR (d) Colored using SVM

Figure 3. Sample coloring of a rotated village

are all features that depend directly on luminance values. Therefore, since the same pixel in both images would produce very different feature vectors, it is not unreasonable to expect that our machine learning models would struggle with this task.[1]

Due to the ill-posed nature of the problem, it is difficult to design reasonable error metrics. For example, whether an image is aesthetically pleasing or not is entirely subjective, and thus difficult to quantify. Consequently, this makes parameter optimization a challenging task. Under normal circumstances, model parameters are learned through a process called cross-validation (Bishop, 2006). During this process, the training set is partitioned into a smaller training set and a validation set. When values for the parameters are selected, the performance of the model can then be measured against the validation set. This procedure allows us to learn near-optimal values for our parameters since it provides an unbiased estimate of the generalization error. In our case, since we have no concrete measure of error, we cannot automatically assess the selection of our parameters. Therefore, parameter tuning was done manually on a per image basis, which we believe played a significant role in all of the errors described above.

7. Conclusions

In this project, we have explored how machine learning techniques can be applied to the colorization process of grayscale images. In particular, we have looked at two types of models: linear logistic regression and support vector machines. Moreover, we discussed various types of errors that occurred during the colorization process of these images. We argued that these errors were caused by localized features and unstructured sampling. Furthermore, we explained that these errors were amplified by the fact that our parameters were all chosen manually. However, despite these difficulties, our methods show promise in accurately coloring various types of images, particularly outdoor scenery. In regards to image transformations, we discussed why our two methods were robust against image rotations but not changes in brightness. More specifically, our models are robust against image rotations because a large portion of our features are derived from SURF descriptors. On the other hand, our models struggle with changes in brightness because this transformation produces significantly different feature vectors.

In terms of future work, there are many avenues that one could explore. For example, it would be worth investigating if different classifiers can produce better results. Algorithms such as nearest neighbors and decision trees are simple classifiers that tend to do well in practice. Though in order to make effective use of these classifiers for image colorization, probability estimates for each color class is required. One potential way to obtain these estimates for nearest neighbors would be to use the Voronoi diagram as the decision boundary. Then the distance between a new data point and that boundary could be a proxy for probability. There has also been some work done

[1] Additional examples can be found in Appendix A.

in obtaining accurate probability estimates for the case of decision trees (Zadrozny and Elkan, 2001). Improvements could also be made with regards to the types of features that we extract. Obtaining global features could help reduce coloring errors that are caused by regions sharing many local similarities. For example, one could perform a region segmentation step in order to extract global information regarding the different areas of an image. Not only would this provide global features, but it would also help with the sampling step. Instead of randomly selecting our training pixels, we could sample a subset of the pixels from every region of the image. This would likely provide a more accurate characterization of the different colors and textures within the image.

On a different note, it might be worth investigating how the performance of our algorithms could be improved if multiple training images are used. This framework would also generalize nicely to film colorization since contiguous movie frames are all similar to one another. In such a setting, the goal would be to color the scenes of a black and white movie. The training set would consist of the first few frames of a particular scene, and these colored frames would then serve as a basis for automatically coloring the remaining frames of that scene. Finally, we believe that it would be worthwhile to develop concrete error metrics for the image colorization problem. This would facilitate the task of automatically learning model parameters, which would likely reduce many of the errors that arise during the colorization process of grayscale images.

Acknowledgments

We would like to thank Professor Dan Lizotte for his helpful suggestions during the course of this project. His feedback is greatly appreciated.

References

M. Ashikhmin, K. Mueller, and T. Welsh. Transferring Color to Greyscale Images. *ACM Trans. Graph.*, 21 (3):277–280, 2002.

H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *ECCV*, pages 404 – 417, 2006.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006. ISBN 0387310738.

Y. Boykov and O. Veksler. Graph Cuts in Vision and Graphics: Theories and Applications, 2006.

G. Charpiat, I. Bezrukov, Y. Altun, M. Hofmann, and B. Schölkopf. Machine Learning Methods for Automatic Image Colorization. In *Computational Photography: Methods and Applications*, pages 395 – 418. CRC Press, 2010.

A. Delong, A. Osokin, H.N. Isack, and Y. Boykov. Fast Approximate Energy Minimization With Label Costs. *International Journal of Computer Vision*, 96(1):1–27, 2012.

R.W.G. Hunt. *The Reproduction of Colour*. Wiley, 2005. ISBN 9780470024263.

H. Jégou, M. Douze, and C. Schmid. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, pages 304–317, 2008. ISBN 978-3-540-88681-5.

A. Levin, D. Lischinski, and Y. Weiss. Colorization Using Optimization. In *ACM SIGGRAPH 2004 Papers*, pages 689–694. ACM, 2004.

S. Liu and X. Zhang. Automatic Grayscale Image Colorization Using Histogram Regression. *Pattern Recognition Letters*, 33(13):1673 – 1681, 2012.

F. Pitié, A. Kokaram, and R. Dahyot. Enhancement of Digital Photographs Using Color Transfer Techniques. In *Single-Sensor Imaging: Methods and Applications for Digital Cameras*, pages 295 – 321. CRC Press, 2008.

L. Velho, A. C. Frery, and J. Gomes. *Image Processing for Computer Graphics and Vision*. Springer Publishing Company, Incorporated, 2nd edition, 2008. ISBN 1848001924, 9781848001923.

C. Wang, N. Komodakis, and N. Paragios. Markov Random Field Modeling, Inference and Learning in Computer Vision and Image Understanding: A Survey. *Computer Vision and Image Understanding*, 117(11):1610 – 1627, 2013.

B. Zadrozny and C. Elkan. Obtaining Calibrated Probability Estimates from Decision Trees and Naive Bayesian Classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 609–616. Morgan Kaufmann, 2001.

A. Colorization Examples

(a) Training image (b) Test image

(c) Colored using LR (d) Colored using SVM

Figure 4. Sample coloring of a fish under water

(a) Training image (b) Test image

(c) Colored using LR (d) Colored using SVM

Figure 6. Sample coloring of a small food tray

(a) Training image (b) Test image

(c) Colored using LR (d) Colored using SVM

Figure 5. Sample coloring of small icebergs

(a) Training image (b) Test image

(c) Colored using LR (d) Colored using SVM

Figure 7. Sample coloring of desert rocks