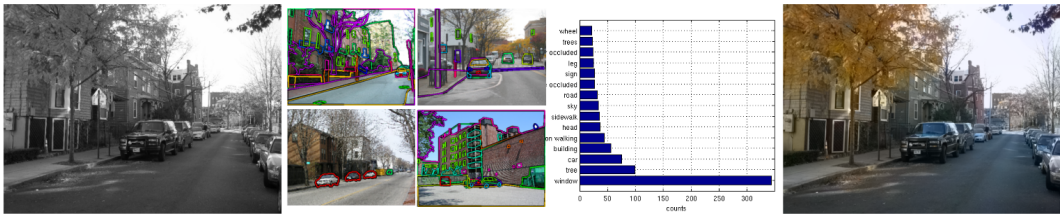
Auto-colorization Exploiting Annotated Dataset

Sungmin Lee
Department of Computer Science
Brown University
sungmin@cs.brown.edu



Abstract

Colorization is a very challenging task for computers which requires very high performance of segmentation, object recognition, color understanding, etc., and even for humans, it's very demanding and arduous work to fully accomplish. To achieve this task, there traditionally has been three approaches: example based model, scribble based model, and data-driven based model which introduced relatively recently, but none of those algorithms are human-intervention free, nor generically work for any arbitrary images.

In this paper, I introduce a data-driven based auto-colorization algorithm which is applicable for an arbitrary query image without extra human labors. I set a hypothesis that if you have many similar images to a query image, a query image is highly likely to have the same objects as in the similar images. To prove this, I exploited the biggest annotated dataset in the world, LabelMe [12], and built a statistical model estimating likelihood of objects in the query image. Once I have a probabilistic map, I simply transfer the colors to each map by using filling algorithm introduced by Levin et al. [4]

1. Introduction

Colorization is a task mapping plausible colors on regions of a grayscale image and this technique has been used in many areas for various purposes. In contradiction to its relatively simple concept per se, automatic colorization requires very high level performances on image segmentation, object detection, and even perceptual color understanding to lead a good result, and it is obvious that better algorithms of these fields you use yield more convincing results you will get.

Since colorization is a very challenging and sensitive task to be automated, a dominant number of researches have focused on "minimizing" human labors rather than fully automatized color generation. Due to this trend derived by the difficulty of colorization, there has been two mainstreams on colorization, which are example-based [14, 6] and scribble-based [4, 9] approaches, and thanks to rapidly growing the enormous number of web image data, researchers have recently started to utilize large image datasets such as Flickr, Google Image Search and investigate those previous models on them to yield better results with less efforts [1, 7]. I will, from now on, address this approach as a data-driven model in this paper.

Unfortunately, Each of these models has their own drawbacks respectively. For instance, Example-based models require a very similar scene to a query image so that the query image can leverage the color maps [14] or textures [6] of the similar image. Although this algorithms are beneficial when you want to apply one source image to many query images, i.e., sequential movie frames in grayscale, finding a similar scene can be a arduous work for users and also the results from example-based models are generally less convincing than scribble-based models. Scribble-based model proposed by Levin et al. [4] shows surprisingly good results in comparison to example-based models, but these algorithms require a lot of human labors since users should choose colors and stroke the regions they want by themselves. There has been researches to simplify this task such as in [9], selecting a right color for a right object is still not a trivial work which requires users good senses of painting. As you can see, building a fully annotated colorization algorithm is non-trivial work at all.

In this paper, I propose a statistical approach for a fully automated colorization algorithm for an arbitrary image, The key idea of my method is to exploit a fully-annotated

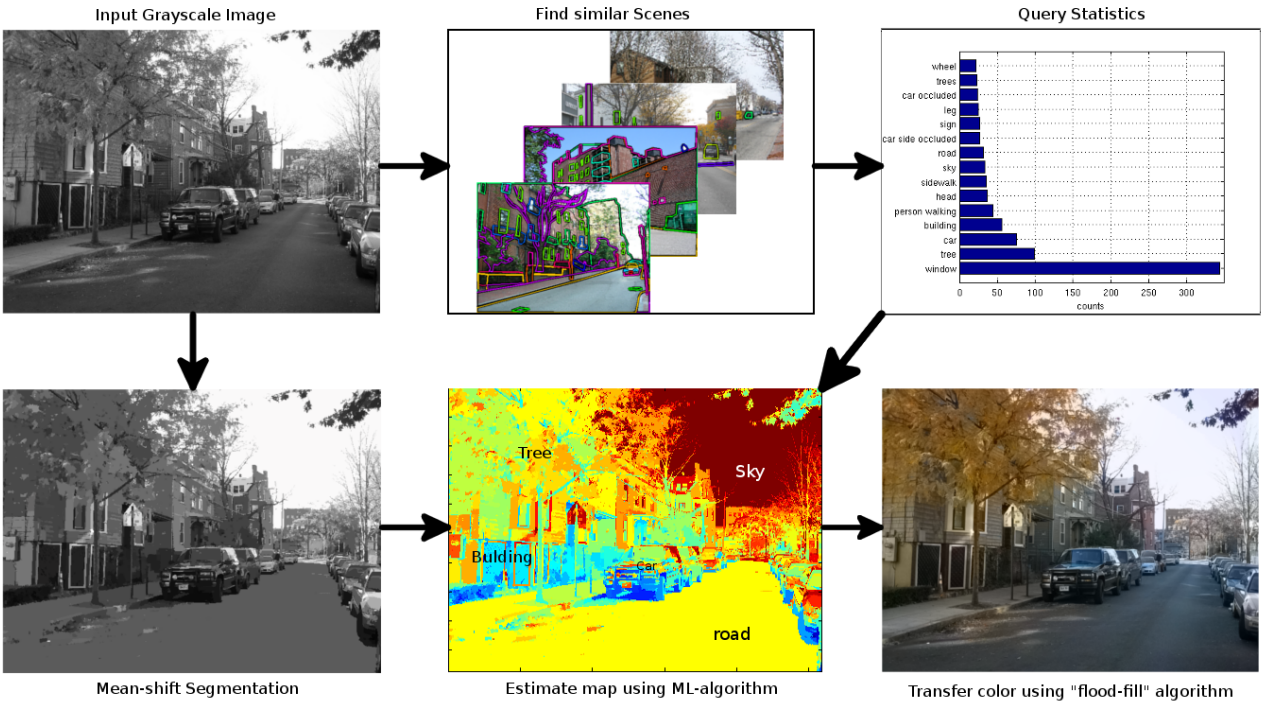


Figure 1. Given grayscale image, the algorithm retrieves top-n similar images(middle-top), and from these images it collects the statistics of all the objects(right-top), meanwhile grayscale image is segmented using mean-shift algorithm(left bottom) and from these segments and statistical data, it estimates each likely region(middle-bottom), and finally it transfers colors of objects to corresponding regions(right-bottom)

dataset to induce the colors of objects in a query image. LabelMe by Russel et al. [12] is still growing annotated dataset currently contains more than 50k images and their annotations, and the main advantage of exploiting this dataset for colorization is that you can build a statistical model for your query image by exploring similar scenes. From all the object annotations from similar scenes, I employ a maximum-likelihood algorithm to predict objects from a query image based on a position, width, height and size of each segment. A user will retrieve top-n colorized results in descending order of their confidences in the end.

2. Overview

My algorithm can be divided into three phases. 1) Image collection and build statistical data: Download and clean LabelMe dataset, retrieve similar images to a query image, and collect the features(label, width, height, position and size) of all the objects. 2) Segment the query image and predict a label of each segment: Apply mean-shift segmentation algorithm to get super pixels of regions, and calculate the maximum-likelihood of each region corresponding to object statistics from step 1. 3) Colorization: colorize each region using flood-filling algorithm proposed by Levin et al.[4] and retrieve the results to the user. Figure 1 briefly demonstrates a system flow of this algorithm.

3. Image collection and statistics

3.1. Collecting annotated images

If you imagine a car in your mind, what color would you expect for that? Or if you imagine a t-shirt in your mind, what color would you expect for the shirt? These kinds of objects are generally less sensitive to their colors since they have a lot of variations. Even if you see some unlike colors, say green or pink, you would think it is, at least, possible but not “unreal”. What about skies, roads, bushes, or even stop signs? You would think that the colors are very awkward if they are not in the colors what they are supposed to be in. This is why object detection is essential for an automatic colorization task.

Exploiting an annotated dataset possibly brings a lot of advantages as long as the annotations are reliably correct and one of those advantages is an object detection itself. LabelMe dataset is a large and growing annotated dataset proposed by Russel et al. [12]. Even though it may not be large enough to cover all the kinds of arbitrary query images for auto-colorization, it is still obvious that it is currently the

Figure 2. Retrieved similar images to a query image. Images in red boxes are falsely retrieved images, but this images do not critically harm the final results since a lot of objects from similar scenes are penalizing outliers.

biggest and most reliable annotated dataset in the world.

I downloaded the full LabelMe dataset from the internet, and manually cleaned the data by getting rid of redundant images which are sequential images, grayscale images, panorama images, too small images, and unrecognizable images by human. After this task, I finally had 34,408 clean images of both indoor and outdoor scenes.

3.2. Object statistics from similar scenes

Hypothetically, similar images likely share similar objects in their scenes. For example, if you have a set of street scenes, you would probably have buildings, cars, people, roads, etc. in common, and if you have a set of beach scenes, there would be oceans, sands, big portion of skies, etc. Thus, we can assume that the more similar pictures you have, the more likely you will have the same objects in the query image statistically. In this purpose, I retrieve n-similar images ($n = 100$ in this paper) to estimate the maximum likelihood of segments from the query image.

From many researches, it has been proved that you can find similar images to an input image even without using sophisticated image descriptors [5, 13]. Since I focus especially more on outdoor scenes than indoors or specific object images, I employ gist scene descriptor [10] which performs exceptionally well at scene recognition. I downsampled images to 128x128 and set the orientations and number of blocks to 8 and 4 respectively. Once I have all the gist values from training set, I calculate simple Euclidean distances from a query image and retrieved top-100 closest images in ascending order and those images are considered as similar scenes to the query image.

Figure 2 shows an example of similar images to a query images and their label statistics. There are some falsely retrieved images(in red boxes) but these do not harm the final result since the objects from the wrong images are statistically outliers. Due to different annotation styles from various users, I concatenate the possible synonyms or hyponyms (i.e., person = {walking pedestrian, man, woman, ...}, car = {sedan, SUV, van, occluded car, ...}, etc.) for the final results. For each label, I assigned four features: widths, heights, sizes and positions. For position feature, I only consider three vertical positions: top, middle and bottom. This actually improves the precision significantly since vertical position represents a rough depth map of objects. For example, if "sky" is predicted from dataset, it would be never located at anywhere but the top, and in this case, even if you see a person-like segment at the top of a query image, it is highly unlikely to be a actual segment of a person.

4. Image segmentation and estimation

4.1. Image segmentation

Different from the training set from LabelMe, a query image doesn't contain any annotation information in it. The optimal way to generate partial objects in an arbitrary image without any human intervention is to apply image segmentation algorithm. If we suppose that we have perfect segmentations in an image, it means each segment will represent one object. Since we have a statistical object information of similar scenes from previous section, we are presumably able to predict each segment. Current state-of-art image segmentation algorithms are somewhat still far from perfection and considered as one of the hardest tasks in computer vision field, however there has been vigorous researches on this topic, and there are a few techniques I can migrate for the colorization task.

Boykov et al.[2] and Rother et al.[11] proposed graph-cut models which are the state-of-art object segmentation algorithm, but these algorithms require some human interaction to select a right region for an object, and this aspect discourages to be employed for an automatized algorithm despite the extraordinary performances themselves.

Comnicu and Meer [3] introduced Mean-shift algorithm and this algorithm converges neighboring pixels which have similar intensity to the base pixel without any human intervention. By applying region fusions to these converged regions iteratively, you can obtain the fully segmented result in the end. There are some advantages using this algorithm: 1) Mean-shift algorithm uses L*u*v color space which consists of one luminance channel and two chromatic channels, so this algorithm is not affected by color channels to make a good segmentation. 2) Since mean-shift segmentation yields converged regions only based on luminance, it is very convenient to directly transfer the colors one to another. 3) This algorithm is relatively fast. Figure 3 shows an original image(left), segmented image(right) respectively.

Figure 3. Mean-shift segmentation example: An input image(left) is segmented by mean-shift algorithm(right) similar pixels are converged into superpixels and make segmented regions

4.2. Statistical object estimation

Given the hypotheses that segmentation result from previous section is tolerably reliable and that there are generally the same objects(labels) in a query image as in similar images, it is possible to predict what each segment would be using some statistical/machine learning methods.

Maximum-likelihood is somewhat old and primitive algorithm among those techniques but still vigorously used for a variety of statistical problems in computer science such as natural language processing thanks to its simple and straight-forward concept.

From machine learning perspective, Maximum-likelihood algorithm is considered as a supervised algorithm, which maximizes a set of combination of each probability fields, generally defined as,

$$\hat{\alpha} = \arg\max_{\alpha} \; L_l(\alpha) \tag{1}$$

where $L_l(\alpha)$ is a likelihood function and this term in this algorithm indicates that "Find a combination of probabilities of features, and what is the label of segment when this combination is the maximum?" I set four different features to define a likelihood function for label, and likelihood function $L_l(\alpha)$ will be a multiplication of each feature. In this paper I define function $L_l(\alpha)$ as,

$$\begin{aligned} L_l(\alpha) = \prod (\; & P(x_{label} \,|\, x_{pos}) \\ & * P(x_{size} \,|\, x_{label}, x_{pos}) \\ & * P(x_{width} \,|\, x_{label}, x_{pos}) \\ & * P(x_{height} \,|\, x_{label}, x_{pos})) \end{aligned} \tag{2}$$

, where $P(x_{label} \,|\, x_{pos})$ is a probability that a $label$ will appear in specific location at specific pos, and this distribution is defined as, where $n(label_x)$ denotes the number of total $label_x$, k denotes the number of positions ($k = 3$ in this paper since there are only three positions: top, middle and bottom), and α denotes a small jitter to prevent from zero probability (α = 0.01 in this paper).

For the rest of equations for size, width and height, I use a different approach since all of those features are continuous, not discrete. As a typical method, I leverage Gaussian distribution. I first calculate the mean and deviation of $\{size, width, height\}$ of each label, such as $label_x \in \{car, person, building, road, \dots\}$ in each position $pos \in \{top, middle, bottom\}$ and I assign a probability as,

$$p(feat \,|\, pos) = \begin{cases} 0.6 & \text{if } -1\sigma < \mu < 1\sigma \\ 0.3 & \text{if } -2\sigma < \mu < 2\sigma \\ 0.1 & \text{else} \end{cases} \tag{3}$$

Thus, the maximum-likelihood of the total equation $L_l(\alpha)$ is a task to find the most probable object for a given segment from its position, size, width and height. This yields an estimation map of the query image and we are just ready to transfer the color information.

5. Colorize the regions

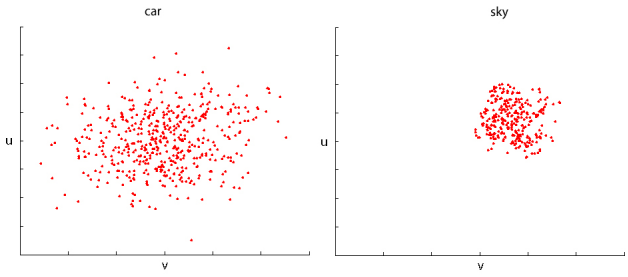


Figure 4. An example of color distributions of car and sky in $u * v$ colorspace. car (left) has more sparse distribution than sky (right) For this reason, if you take $\pm 1\sigma$ deviation you will possibly obtain general color variations

Now, I have an estimation map of the query image from the previous section, and I can basically transfer the according colors to the regions. Of course it is possible to simply fill the color in the region, but there are a few possible problem. 1) Mean-shift is not a perfect segmentation algorithm: Mean-shift algorithm sometimes over-segment regions (i.e., parts of buildings are merged to sky, bushes merge a background, ...), and sometimes under-segment regions (i.e., one car is deformed in to many parts,...), thus simple color transfer can lead bad results. 2) Deviations of colors for objects are significantly different: If we take only one color from one specific object, we would, for instance, see only black cars, but, on the other hand, if we randomly pick a color from an object set, we might loose a consistency of colorization (i.e., Some part of sky is blue and some part of sky is dark or red)

For this reason, I calculate the deviation of color distribution for each object, and take $\mu \pm 1\sigma$ of color distribution for an object. Figure 4 shows a different color distribution between "sky" and "car" objects.

To also tackle the first problem, I employ a variation of "flood-filling" algorithm which is proposed by Levin et al. [4]. This algorithm basically compares a pixels $p(x, y)$ to its neighboring pixels and transfer the color of p to its neighbors as long as the intensity doesn't change significantly. This algorithm is based on the hypothesis that the intensity changes significantly generally when different objects are neighboring.

Levin's algorithm [4] has some advantages on my project. 1) The one big disadvantage of this algorithm is that it needs a lot of human scribbles especially when a scene is complicated. But my algorithm provides all the segment information and extracting some color points from the related objects is a trivial task. 2) As the second disadvantage of the algorithm, choosing a right color for a right region is not a easy task for non-artistic people. By applying my algorithm to Levin's it automatically creates a palette from real scenes. 3) This algorithm blends colors very naturally at the edge of different regions. It yields more convincing results than just transferring a color to a segmented region.

6. Results

I build five different categories for a test set to evaluate the results, which are coastal scene, urban street, indoor scene, person oriented, and object oriented. Figure 5 shows colorization results. As you can see from these results, this algorithm performs well on general outdoor scenes, but shows very poor results at indoor or object oriented scenes. This drawback possibly comes from 1) lack of data, 2) characteristic of gist descriptor, 3) or complexity of indoor scenes. Especially, based on the object statistics, we can easily assume that my algorithm tends to consider all the scenes as outdoor scenes. The possible resolutions of these weakness will be discussed in following discussion section.

7. Discussion

I this paper, I introduced a fully automated colorization algorithm using annotated dataset. The main contributions of this paper are 1) that this is the first paper which provides fully automated colorization algorithm for an arbitrary query image, 2) that this paper shows a probabilistic approach for a colorization, and 3) that this paper shows a practical and successful application of LabelMe dataset suppressing poor-annotation-noises with statistics

The result shows relatively convincing results but there are still a lot of rooms to improve this algorithm as well. Above all, this paper doesn't give an evaluation data. This happened because there's no absolute ground-truth for colorization. Also, since there's no automatic colorization algorithm, it can be somewhat unfair to compare the results to others. The only possible evaluation is using human annotation such as Amazon Mechanical Turk, asking the people to evaluate how much convincing these results are.

To improve indoor, or object colorization, I could suggest a couple of possible solutions. One of them is a size of dataset. As it is proved in [5, 13], the results will significantly improve as the dataset grows. Second solution is using other image descriptors such as SIFT [8] which performs superbly well at feature comparisons. Since gist is designed to outdoor scene rather than limited scenes, using other descriptors or combining multiple descriptors will help improve the results.

As an initial model of arbitrary image colorization, I believe auto-colorization task is somewhat related/similar to statistical AI model such as natural language processing in terms of a fact that there are no absolute ground-truth but statistically plausible results. I expect tons of annotated data which are built up and well trained statistical model will help reach human-level of colorization task in the near future.

References

[1]

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, September 2004.

[3] D. Comaniciu, P. Meer, and S. Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[4] A. L. Dani, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23:689–694, 2004.

[5] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), 2007.

[6] R. Ironi, D. Cohen-Or, and D. Lischinski. Colorization by example. In O. Deussen, A. Keller, K. Bala, P. Dutr, D. W. Fellner, and S. N. Spencer, editors, *Proceedings of the Eurographics Symposium on Rendering Techniques, Konstanz, Germany, June 29 - July 1, 2005*, pages 201–210. Eurographics Association, 2005.

[7] X. Liu, L. Wan, Y. Qu, T.-T. Wong, S. Lin, C.-S. Leung, and P.-A. Heng. Intrinsic colorization. *ACM Trans. Graph.*, 27(5):152:1–152:9, Dec. 2008.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[9] Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum. Natural image colorization. In J. Kautz and

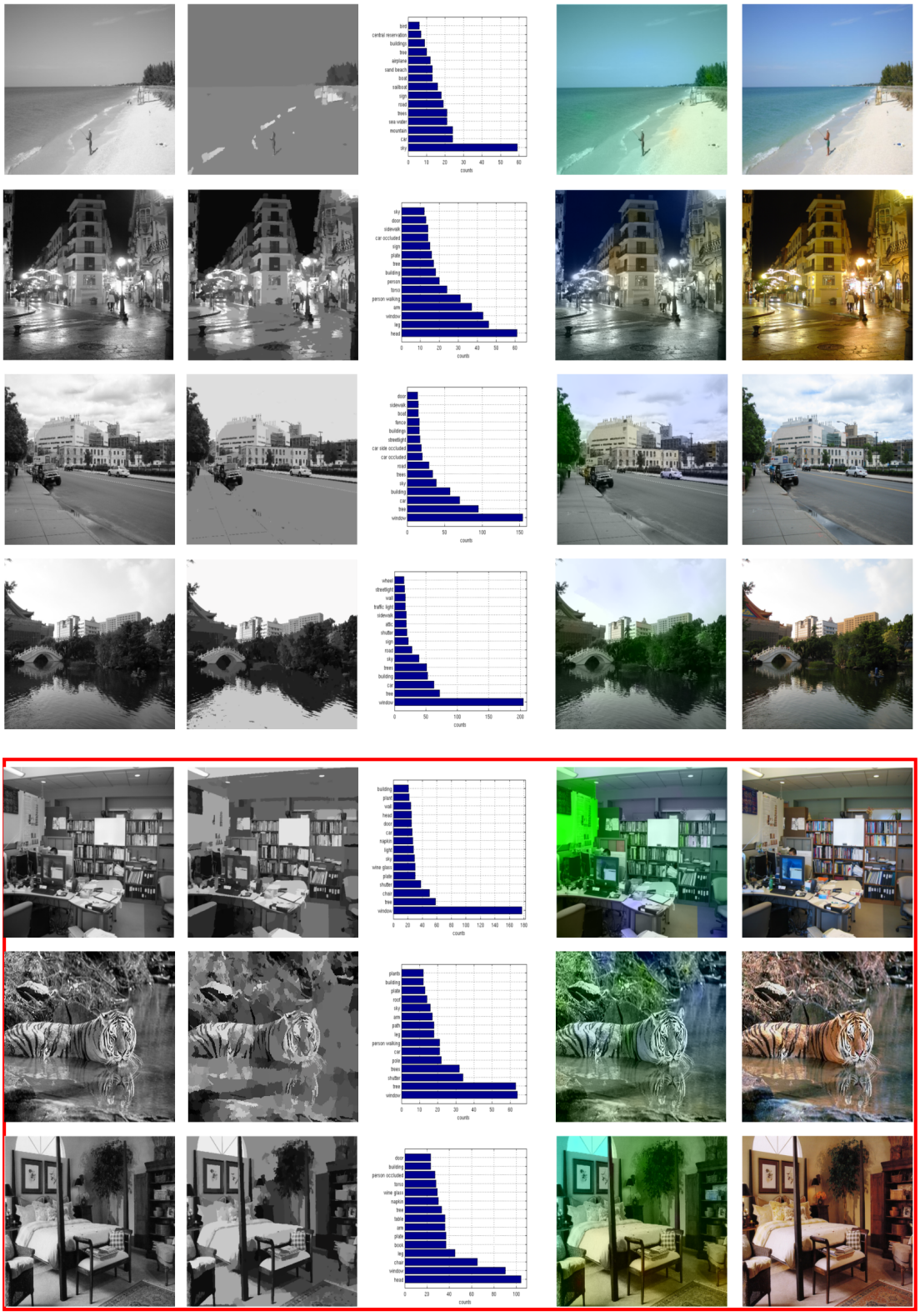


Figure 5. This shows the results of algorithm. from the left, query image, segmented image, object statistics(before merge), my results, and ground-truth. The last three rows in a red box indicates are awfully colorized results. As you see, this algorithm shows a significant weakness at indoor, object oriented scenes.

S. Pattanaik, editors, *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*. Eurographics, June 2007.

[10] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. In *Progress in Brain Research*, page 2006, 2006.

[11] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, Aug. 2004.

[12] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: A database and web-based tool for image annotation. *Int. J. Comput. Vision*, 77(1-3):157–173, May 2008.

[13] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, Nov. 2008.

[14] T. Welsh, M. Ashikhmin, and K. Mueller. Transferring color to greyscale images. *ACM Trans. Graph.*, 21(3):277–280, July 2002.