Violet
Optimal Image Selection with Machine Learning
CS231A Final Report

Fahim Dalvi
fdalvi@cs.stanford.edu

Kai-Yuan Neo
kneo@stanford.edu

March 18th, 2015

1 Introduction

People often capture several photos of the same scene to produce the best image. Manually choosing the best image out of the candidates is a time consuming process. We propose an algorithm to automatically detect the optimal image out of a set of candidate images. This eliminates the need for people to spend time evaluating the quality of their images, and allows them to focus on enjoying the memories they have experienced.

Figure 1: Poorly-lit

Figure 2: Well lit and focused

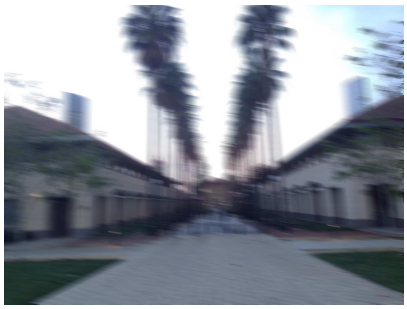

Figure 3: Poorly-focused and blurred

2 Previous work

2.1 Review of previous work

Past work has focused on optimizing small subsets of the features and techniques tested in Violet. Apple has built a similar but simpler optimal image classifier for IOS 8, that classifies based on facial features and blur detection. Unfortunately, they have not released the actual algorithms behind their classification. From our limited testing, it generally avoids blurred images, but does not always pick the clearest image. Other papers that we have come across concentrate on specific features in an image. For example, researchers in Ireland published a paper on detecting image orientation in 2006[8] with the goal of quickly detecting slanted images. Another paper by Mai et al. uses the concept of detecting the rule of thirds in photos[10]. There has also been work done on finding the most representative image from a set of images by Chu and Lin[3]. We also found several papers that aim to detect reflection and colorfulness of images, albeit not for finding the best image in a set. Finally, a paper by Li, Loui and Chen of a very similar goal as ours describes algorithms to get an aesthetic score for an image[9]. The algorithm they describe heavily focuses on faces in images.

We did not find any published research that has combined multiple such features to classify images. Violet seeks to accomplish this by combining many of the above algorithms and others to get a holistic score for each image.

2.2 Key contributions of project

Violet contributes feature selection research and annotated data to the automatic image rating community.

Regarding feature selection research, Violet demonstrates a case where combining 8 most commonly used features to determine image quality successfully predicts the best image from a set of images. These 8 features include blurriness, brightness, vibrancy, reflection, colorfulness, fraction of eyes closed, facial blur, and image composition. Each of these features has been the subject of extensive research for feature extraction techniques, and Violet uses and builds on top of them, producing novel feature extraction techniques and results.

As examples, Violet uses OpenCV to detect faces and eyes, but combines that with existing blurred image research to produce a facial blur score for each image. The study of image composition has been largely subjective until recently, and Violet seeks to score conformance with the Rule of Thirds with a novel image composition feature extraction technique. Violet also presents a new algorithm to detect closed eyes in very low resolution images.

Regarding data tagging, in the process of completing the project we tagged 5000 images as good or bad. These images contain various scenes and setups including lakesides, people, camping, weddings, cars, flowers, and other miscellaneous images. These positive and negative tags can be used as-is by future researchers seeking to determine quantification of image quality. To simulate good and bad images of the same scenes, we have written algorithms that modify image blurriness, brightness, and vibrancy and generate 'bad' photos from a 'good' photo.

3 Technical solution

3.1 Summary

This project is divided into 3 components:

1. Data collection and labeling
2. Feature selection and extraction
3. Machine learning algorithm selection and implementation

Data labeling and feature extraction took the majority of the time for this project. To speed up the process, we built a web application for easy classification of images as 'good' or 'bad'. The application automatically saved each annotation in a convenient format that could be used to provide class labels for each image.

To avoid overfitting on specific scenes, we decided to tackle the more general problem of giving an image a score, and then applying the results to our specific case of consumer photos of the same scene. Hence, our dataset comprises many images each from different scenes. To generate test data to evaluate image selection from images of the same scene, we wrote an algorithm to randomly adjust blur, brightness, and exposure of an image to generate bad images.

Feature extraction required extensive research to find appropriate methods to extract the features most relevant in determining image quality. The initial implementation of Violet (baseline) utilized 2 features exclusively: blurriness and brightness. The baseline suffered from over-generalization of images, and resulted in linearly inseparable data. The final implementation of Violet expanded on this to include 8 features, each of which accurately discriminated between good and bad images based on their metric.

3.2 Datasets

After searching through several vision dataset indexes, we decided to use the **CERTH Blurred Images Dataset**[2], as it was the most appropriate dataset for the purposes of our project. This dataset contains

about 1000 undistorted images, and 1200 blurred images. This set is helpful because blurriness is one of the strongest contributing factors to bad photographs. Although we could classify the blurred images as bad photos quickly, this was not the same for the undistorted images. We had to manually tag them as good or bad. A positive outcome of this manual tagging process was that we were able to use this dataset to test how light-levels affect classification of images.

After getting some initial results from the CERTH dataset, it proved to be insufficient for our needs. This was because it had a larger number of 'bad' photographs, and hence our model was performing poorly on the 'good' photograph class. Moreover, the CERTH dataset did not fully represent our use case, which was targeting consumer photos. Hence, we decided to augment our dataset with images downloaded from ImageNet[7], an online database maintained by the Stanford Vision Lab. We downloaded 4000 images from ImageNet to diversify and normalize our dataset. Some of the categories we used included camping, weddings, cars and people. As was the case with the CERTH dataset, not all of these images were 'good' photographs, and hence we had to manually annotate these.

Some other datasets that we did not use were **Indoor Scenes with Various Lighting Conditions 1**[11] and **Indoor Scenes with Various Lighting Conditions 2**[4]. The former contains images of several indoor scenes at varying illumination, but the image subjects and backgrounds were too artificial. The latter contained images whose lighting conditions were more artificial than those in the types of images we wished to evaluate.

3.3 Feature Extraction

3.3.1 Brightness

The first feature we extracted was brightness, as it is one of the most basic factors in determining the quality of a photograph. To achieve a qualitative measure of brightness, we used relative luminance, as it incorporates information such as what colors the human eye perceives more sharply. For each pixel, we compute the luminance as:

$$luminance = 0.2126 \times R + 0.7152 \times G + 0.0722 \times B,$$

where R, G and B denote the red, green and blue components of the pixel respectively. We then compute the average luminance over all the pixels in the image.

As expected, relative luminance cannot be linearly separated into 'good' and 'bad' photograph classes, as 'bad' photographs can either be too dark or too bright. After looking at the data we had, we found the average luminance for 'good' photographs, and then considered the absolute distance from this average luminance as our score for brightness. Hence, 'good' photographs would therefore have a much smaller score, while both dark and overly bright images would have a higher score.

3.3.2 Blurriness

The next feature we tackled was blurriness. We studied several approaches such as Haar wavelet transforms or measuring the width of edges in an image. In the end, we used the approach used proposed by Su, Lu and Tan[13]. The premise of the algorithm is that blurred images have a lot more information about *coarse-grained details* in an image, while focused images also store information about *fine-grained* details. The algorithm also uses the fact that when we decompose an image using SVD, the first few singular values incorporate information about the *coarse-grained details*, while the last few singular values incorporate information about the *fine-grained* details. Therefore, we consider the ratio

$$\beta = \frac{\sum_{i=0}^{t} \lambda_i}{\sum_{i=0}^{n} \lambda_i},$$

where n is the total number of singular values and t is how many singular values we want to consider. We choose t to be $\frac{1}{6}^{th}$ of n. Hence, β will be higher for blurred images, since *fine-grained* details are missing in a blurred image, the last few singular values will encode less information and be much smaller. One of the biggest advantages of this approach over other ones is that we can detect both focus and motion blur using the singular values.

3.3.3 Reflection

After analyzing several 'bad' images in our dataset, we noticed one common theme was reflections. One common example of this is when photographs are taken from behind a semi-reflective surface, such as window panes. An instance of this can be seen in figures 4 and 5. We decided to implement the algorithm proposed by Ahmed, Pitie and Kokaram[1]. They use the concept of Color Channel Independence to determine the level of reflectance in an image. Fundamentally, they try to separate the individual color channels into multiple layers, and the degree to which they can successfully do this is their score for reflectance. A qualitative measure of this degree of separation is termed as *Generalized Normalized Grayscale Correlation*, or GNGC. Specifically, GNGC can be computed by[12]:

$$GNGC(f,g) = \frac{\sum_{i=1}^{N} C_i^2(f,g)}{\sum_{i=1}^{N} V_i(f) \cdot V_i(g)},$$

where C is the covariance and V is the variance of the respective inputs. As Ahmed, Pitie and Kokaram suggest, we use the Red and Blue channels of an image as inputs the the GNGC function. The higher the value of GNGC, the lower the reflectance in an image.

Figure 4: Example with no reflection

Figure 5: Example with reflection

3.3.4 Vibrancy

Vibrancy of an image was the next factor we used as a feature. This feature was inspired by the histogram equalization technique generally used to sharpen and increase the contrast in an image. In simple terms, the less vibrant an image, the more faded out it looks. To get a qualitative measure for this feature, we decided to use the entropy of the luminance histogram of an image. Generally, images that are faded out tend to have a peak at certain values of luminance, rather than distributed values for luminance. Figure 6 shows examples of vibrant and faded images.

3.3.5 Saturation

Professional photographers always suggest using the HSV colorspace instead of the RGB space, as the channels defined by HSV better separate the various components of an image. Namely, Hue defines the color of a pixel, Saturation defines the "amount" of color, and V defines the lightness of the pixel. We analyzed several features and their combinations like entropy of saturation, average saturation, functions of saturation and lightness combined etc. After our analysis, we did not find any of these particularly helpful in distinguishing between the classes. This feature did not provide any significant information that was different from other features, and hence we opted to not use this as a feature to simplify the system.

3.3.6 Colorfulness

Although the previous feature was not very distinguishing, we still wanted to use the idea of the amount of color in an image as a feature. We implemented the colorfulness metric defined by Haslera and Süsstrunk in their 2003 paper[6]. The metric is defined as follows:

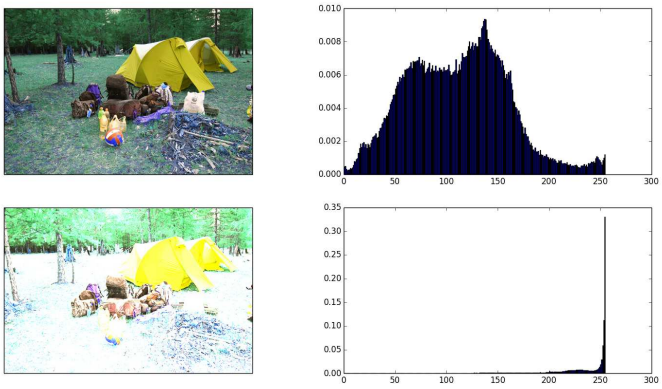

Figure 6: The top row shows a vibrant image along with its luminance histogram, while the bottom row is a less vibrant image

$$C = \sqrt{\sigma_{rg}^2 + \sigma_{yb}^2} + 0.3\sqrt{\mu_{rg}^2 + \mu_{yb}^2},$$

where rg is defined as the difference between the red and the green channels, and yb is defined as the difference between the yellow and the blue channel. The yellow channel is obtained by averaging the red and green values for each pixel.

3.3.7 Facial blur

A major component of consumer photos today is faces, and we found it imperative to make use of facial features in our algorithm. We first use the famous algorithm proposed by Viola-Jones[14] to detect faces in an image. We used an OpenCV implementation to accomplish this. We then apply the blurriness detection on each face, and use the average blurriness for all faces in an image to get a quantitative score for this feature.

3.3.8 Fraction of closed eyes

Using face detection from the previous feature, we then use Haar Cascades to detect eyes in the image. We use a pre-trained model provided by OpenCV to accomplish the eye detection. The aim of this feature was to detect the fraction of eyes that are closed in the image, as photos with open eyes are generally better than photos with closed eyes. Closed eyes are a common occurence when taking group photographs. We studied several algorithms to detect closed eyes such as using HoG features and running a machine learning algorithm to learn the weights. We also analyzed algorithms that used Circular Hough Transforms to detect the iris in the image. Unfortunately, since faces are not usually a large part of pictures, the resolution of eyes was typically low, and the above algorithms did not perform very well. Inspired by several techniques seen in class and other features, we implemented the following algorithm to detect closed eyes in an image:

1. Scale the eye patch to 200x200 pixels. We lose some clarity because of the blurriness, but this helps us achieve better numbers for the steps that follow

2. Perform histogram equalization on the eye patch. This helps us normalize the effects of eyes detected from really bright or dark images

3. Threshold the image such that all pixel values below 50 are dark, and all above are bright. This helps us offset the effects of the lost information from the up scaling of the eye patches

4. Compute the sum of intensities column-wise for the eye patch. Therefore, at the end of this step, we will have a 200 element vector, where the ith element represents the sum of intensities of the ith column in the eye patch

5. Fit a 2nd degree polynomial to the curve obtained from the intensity vector previous step. The intuition behind this step is that if an eye is open, the intensity would increase on either side of the iris (since the iris itself is dark, with surround white areas). On the other hand, for a closed eye, we would not see such peaks.

6. Compute the error between the intensity curve and the fitted polynomial from the previous step. Threshold the error such that values below the threshold indicate a closed eye.

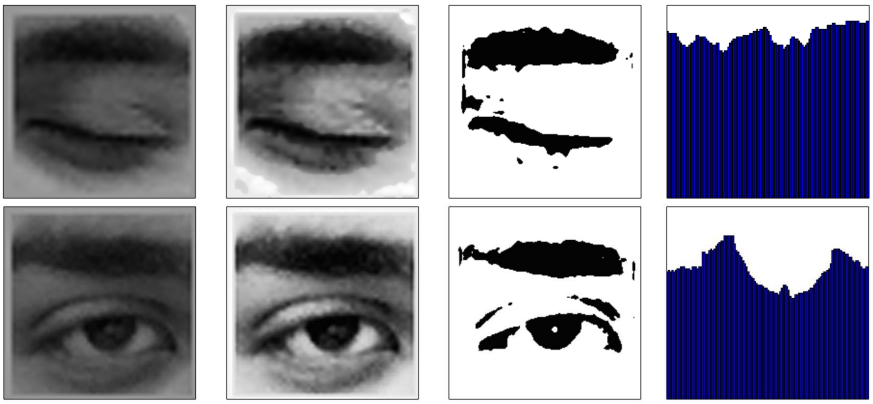

Figure 7: The top row shows the images at different steps of the process. As we can see, the histogram for the closed eye is much flatter than the histogram for the open eye

3.3.9 Image composition

Taking inspiration from a paper by Mai et al.[10], we decided to implement a simpler algorithm to detect the composition of an image due to time constraints. The rule of thirds states that the *subject* of an image should lie on the grid lines created by dividing an image into thirds vertically and horizontally. Figure 8 shows an example of a photograph obeying the rule of thirds.

To get a qualitative measure of the adherence to the rule of thirds, we first scale all the images to a constant size. We then run an edge detector on the image, and count the intensities of the pixels on the edge map around the grid lines. The sum of these intensities is our qualitative measure of adherence to the rule of thirds.

3.4 Algorithms and Machine Learning

3.4.1 Baseline

To serve as a simple baseline, we first used our brightness and blurriness features. We manually set thresholds to classify the good images from the bad ones by analyzing a subset of the dataset.

3.4.2 Naïve Bayes

We implemented an initial machine learning algorithm with Naïve Bayes, assuming all features were independent of one another. This assumption is not true, as shown by the dependence of blurriness and exposure in Figure 9, obtained from empirical results during our testing. The use of machine learning is further justified by the fact that it is difficult to manually tune the classification thresholds due to the dependencies between features.

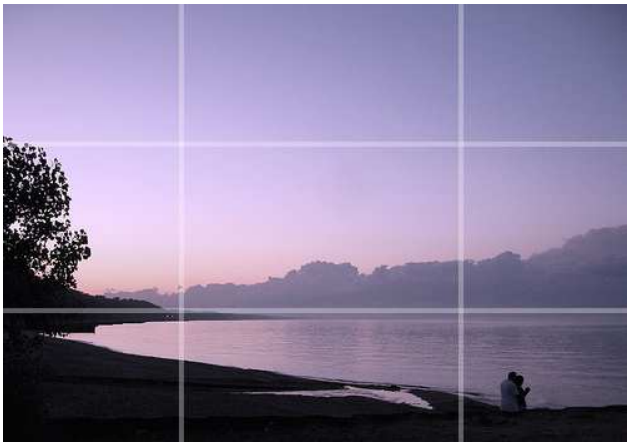

Figure 8: Rule of thirds example

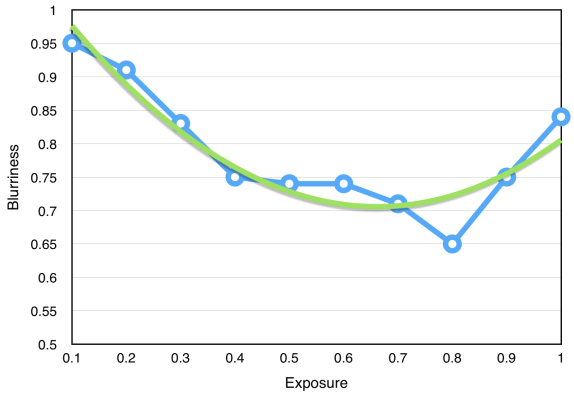

Figure 9: Blurriness vs Exposure

3.4.3 SVM

We moved on to a more complex model by using Support Vector Machines to attain better classification results. We tried several linear and non-linear kernels. After performing a grid-search over various kernels and parameters, we decided to use a linear kernel, as it consistently gave us the best classification accuracy. We faced several issues in this phase of the project. After analyzing the feature space (explained further in experiments section), we realized that our algorithm was not doing very well because our feature space was very clustered, with large overlaps between the classes. We also noticed that some of our features were not linearly separable, and hence the linear kernel was not attaining the highest possible accuracy. Keeping these findings in mind, we started to tune our features and dataset to alleviate these problems.

We first fixed our features so that they were linearly separable. We did this by considering the absolute distance of each feature value to the average good value for that feature. To fix the problems with our data, we significantly expanded our dataset to include images from many more scenes. We specifically made sure that we had images containing faces, reflections and varying levels of color and exposure. We also made sure that the number of examples of each class where roughly the same.

With this, our performance got a significant boost. We continued tuning our parameters to achieve better separating hyper-planes and avoid overfitting.

3.4.4 Logistic Regression

We were still not obtaining the best results, and hence decided to try different classifiers. We noticed that we still had problems with noise: Many 'good' photos would sometimes have feature scores inclining towards the bad class. Since SVM generates a separating hyperplane based on the data points closest to the margin rather than the entire dataset, SVM can be prone to noise when the support vectors are noisy. We posited this occurs in our case, and found that logistic regression performed better. Switching to logistic regression with the same features gave us a small boost of about 1%.

4 Experiments

We performed several experiments varying both the machine learning algorithms and our features. The results from our baseline and logistic regression (the final machine learning algorithm we chose) are described in Tables 1, 2 and 3.

	Bad Images	Good Images
Precision	0.53465	0.62561
Recall	0.80507	0.31733
F1	0.64257	0.42108

Table 1: Baseline Results

	Bad Images	Good Images
Precision	0.66335	0.63453
Recall	0.59917	0.69592
F1	0.62963	0.66381

Table 2: Training Error

	Bad Images	Good Images
Precision	0.69007	0.63906
Recall	0.64480	0.68470
F1	0.66667	0.66109

Table 3: Test Error

As we can see, the machine learning algorithm performs much better, especially on the 'good' class. Our linear SVM implementation boosted our scores by about 2% above Naïve Bayes, and the Logistic regression implementation gave us an additional 1% boost on testing accuracy for both classes. We also tried a quadratic kernel and the RBF kernel as they are non-linear, but neither fit well to our data. We also tried several methods to avoid over-fitting such as setting higher regularization costs and performing early stopping during training. Early stopping was very helpful initially when our dataset was not balanced, but after we expanded our dataset, it did not have much of an effect.

As for feature selection, we assessed each feature individually on a sample set of images, as well as look at the structure of the feature space described by each feature individually and in combination with other features. Figure 10 shows the feature space for three features combined: brightness, blurriness and facial features. As we can see, there is a nice distinction between the green ('good' samples) and red ('bad' samples) areas. As mentioned earlier, we decided to not use the *saturation* feature, as it did not separate well in the feature space.

Analyzing our feature space also helped us resolve several issues we faced early on with poor machine learning performance. We noted from the feature spaces that our features were not very discriminating, and that guided us to improve the existing features and add new ones based on analysis of 'good' and 'bad' photos. We also learned a lot of insight on the data by looking at the feature spaces. For example, we initially noted that our facial feature did not have much of an impact on the feature space, and rightly concluded that our data simply didn't have many faces. This helped us expand our dataset in the right manner.

For our implementation of closed eye detection, we decided to qualitatively test it on an existing dataset. The **Closed Eyes in the Wild** (CEW) [5] dataset gives us about 5000 annotated eye patches with both closed and open eyes. On this dataset, our algorithm has 75% accuracy. Although a higher accuracy would be better, this suffices for the purposes of comparing photos of the same scenes with the same faces. The detailed results are outlined in 4

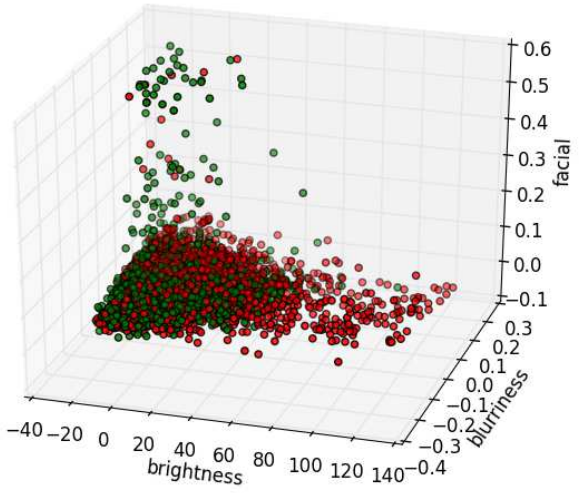

Figure 10: Feature space

	Open Eyes	Closed Eyes
Precision	0.75040	0.74682
Recall	0.75711	0.73993
F1	0.75374	0.74336

Table 4: Accuracy on the CEW dataset

We started out with the goal of accurately selecting the optimal photo in a set of consumer photos from the same scene. During the course of the project, we focused on the general problem of giving an image a score, immaterial of the scene it originated from. Hence, we decided that to really gauge the success of our algorithms, we must test it on the original problem. Therefore, we choose 200 images from different scenes, and synthetically generated bad photos for each scene. We applied a random amount of exposure (in both directions), motion blur and focus blur to generate the bad photos. Hence, at the end of this step, we had 200 sets of photographs, where each set had 5 photographs with various amounts of blurriness and brightness. We used our classifier to rank the images in each set. Our algorithm classified the actual best image in each set as either rank 1 or 2 with **99.5**% accuracy. Also, with an accuracy of **89.5**%, the algorithm gave the highest ranking to the actual best image.

5 Future Work

Our eventual goal is to enable a user to provide as input an entire album of photographs, and receive as output the best image from each scene. To do this, we will need to identify which images are part of the same scene in the album. We can use one of the various features we have learned in class for matching, like SIFT or HOG's to complete this task. We would also like to incorporate more tips from photography experts from around the world, like checking for symmetry and centering a photograph on the subject's dominant eye.

Finally, we would also like to have a web application where consumers can upload albums and we choose the best image from each scene for them. We also envision adding feedback capabilities to the app, so that our algorithms can improve over time as a greater number of people use it.

6 Conclusion

In conclusion, we achieved all what we had sought to achieve except the web application. Although we had some time at the end of the project to work on the web application, we decided to instead improve our features and perform more analysis on the algorithms. We achieved reasonable results given our goals, and we hope to continue work on this project and make the research available soon.

7 References

[1] Mohamed Abdelaziz Ahmed, Francois Pitie, and Anil Kokaram. "Reflection detection in image sequences". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 705–712.

[2] *CERTH Image Blur Dataset*. `http://mklab.iti.gr/project/imageblur`.

[3] Wei-Ta Chu and Chia-Hung Lin. "Automatic selection of representative photo and smart thumbnailing using near-duplicate detection". In: *Proceedings of the 16th ACM international conference on Multimedia*. ACM. 2008, pp. 829–832.

[4] *DTU Robot Image Data Sets*. `http://roboimagedata.compute.dtu.dk/?page_id=24`.

[5] X.Liu F.Song X.Tan and S.Chen. *Eyes Closeness Detection from Still Images with Multi-scale Histograms of Principal Oriented Gradients*. Pattern Recognition, 2014.

[6] David Hasler and Sabine E Suesstrunk. "Measuring colorfulness in natural images". In: *Electronic Imaging 2003*. International Society for Optics and Photonics. 2003, pp. 87–95.

[7] *ImageNet Image Database*. `http://image-net.org`.

[8] Hervé Le Borgne and Noel E O'Connor. "Pre-classification for automatic image orientation". In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. Vol. 2. IEEE. 2006, pp. II–II.

[9] Congcong Li, Alexander C Loui, and Tsuhan Chen. "Towards aesthetics: a photo quality assessment and photo selection system". In: *Proceedings of the international conference on Multimedia*. ACM. 2010, pp. 827–830.

[10] Long Mai et al. "Rule of thirds detection from photograph". In: *Multimedia (ISM), 2011 IEEE International Symposium on*. IEEE. 2011, pp. 91–96.

[11] *Phos Benchmark Image Dataset*. `http://utopia.duth.gr/~dchrisos/pubs/database2.html`.

[12] Bernard Sarel and Michal Irani. "Separating transparent layers through layer information exchange". In: *Computer Vision-ECCV 2004*. Springer, 2004, pp. 328–341.

[13] Bolan Su, Shijian Lu, and Chew Lim Tan. "Blurred image region detection and classification". In: *Proceedings of the 19th ACM international conference on Multimedia*. ACM. 2011, pp. 1397–1400.

[14] Paul Viola and Michael Jones. "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–511.