ISSN: 0976-9102 (ONLINE)
DOI: 10.21917/ijivp.2018.0268

REMOVAL OF UNWANTED OBJECTS FROM IMAGES USING STATISTICS

Sourav Kulkarni
Department of Computer Engineering, Pune Institute of Computer Technology, India
E-mail: souruly@gmail.com

Abstract

Nowadays with cheap digital cameras and the easy availability of camera enabled smartphones, people have been taking a lot of photos. But many a times, it happens that the photos clicked have some unwanted objects appearing in the picture that either partially or completely obscure the subject or their presence spoils the quality of the photo in some or the other way. Modern powerful image editors and in-painting algorithms are capable enough to remove these abnormalities in post-processing to provide a convincing output image. But these often include manual work which makes them time-consuming. Also, these either require an expert user or some complex algorithms for their functioning. The algorithms and methods discussed in this paper aim to provide a much simpler approach to solve these problems using basic statistics. A comparative analysis of their efficiency is also provided.

Keywords:

Image Processing, Statistics, Image Stacking

1. INTRODUCTION

In this modern day and age we hardly find ourselves in a position in which we cannot take a quick picture of something that's in front of us. Digital pictures are so important because they are less volatile than our natural memories formed in our brains. Once generated, they can be preserved indefinitely on the perfect form without loss. A corruption of the image, therefore foils the entire effort. Corruption can be any type, one of which is having an unwanted object/artifact in the image. Take for example, random tourists appearing in your vacation photos which would have looked a lot better without them in it. Sometimes, it may be possible to take an image void of such eye sores that vastly generate the appreciation of an image by the viewer. To do this you may need to perfectly time your shot, positioning the camera such that these objects don't appear in the frame, physically removing the obstructions-if any, etc. But this isn't always possible. Sometimes it is inevitable that some unwanted object comes in the view of the camera and you can do nothing about it. You may have to click images with these objects popping in your photo.

One way to remove these unwanted artifacts from your image is through post-processing. Post-processing is making alterations to an image after its generation. Modern image editors and/or any of the in-painting algorithms can be used to remove the unwanted artifact from the image and reconstruct the image as it should have been if the object had never been there. Manual image reconstruction or the in-painting algorithms basically work in 3 steps – identification of the unwanted object, removal of the unwanted object, filling the empty space thus created with new data that fits well with the rest of the image. The first step of identification of the unwanted object/discrepancy in the image is almost always manual. The area to be considered for correction is highlighted before any work is done on the image. The second step just involves clearing up the pixels that lie in the affected region. The third and the most important step is filling up the empty void created in step two with new data. This is essentially making a clever guess, extrapolating the available knowledge to paint the empty space in a way that best fits the rest of the picture. The available knowledge may be the lifetime experience of an expert photo editor, the characteristics drawn from the image itself, texture synthesis, edge extrapolation or inferences drawn from the dataset for an AI-ML based in-painting algorithm [1]-[6].

The algorithms discussed in this paper, on the other hand, involve no manual highlighting of corrupted area and no guess work. Statistical analysis is conducted over a set of input images to reconstruct an output image free from the unwanted artifact while still preserving the subject in all its glory. The essence of this method is that 'the majority can never be wrong'.

This paper is divided into 10 sections. Section 1 provides an introduction to the removing unwanted objects from images along with related work in that field. Algorithms are proposed and their working is explained in section 2 and 3. Section 4 includes sample input images and results. Section 5 includes analysis of results and efficiency of the algorithms. Section 6 talks about all the variables involved in the working of the algorithm. Section 7 and 8 provides the advantages and limitations of the methods presented in this paper. Section 9 talks about the future scope of improvement. Section 10 presents the conclusion of the paper.

2. METHODOLOGY

2.1 BASIC WORKING

This method takes a set of images as input, performs computations on them and spits out a single image as output. The computations involve applying a particular statistical operation on the set of pixels belonging to each of the input images at a specific location. In the scope of this paper, we are going to look at the basic statistical methods Mean, Mode and Median. We shall also look at a way of improving the mean method by use of clustering.

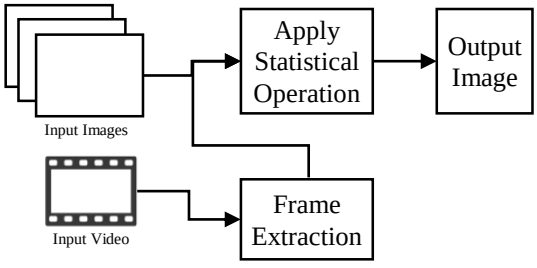


Fig.1. Block diagram explaining the working of the algorithm

2.2 INPUT IMAGES

Before we discuss the algorithm in detail, we shall take a moment to define the type of images that are to be taken as input. The algorithm takes a set of images as input. For the algorithm to work effectively, this set should have the following characteristics.

- The subject of the photograph should remain still across all the images.
- The unwanted object's location with respect to the image origin should be different for each image. Better results are obtained when the locations are distinct and never occupy the same area across different images.

The algorithms mentioned in this paper was tested on a self-generated input images dataset.

2.3 THE ALGORITHM

For every pixel $P_{i,j}$ in the pixel space located at coordinates (i,j)

Step 1: Make an empty set of colors called *C*.

Step 2: Add the color of all the pixels from all images to the set *C*.

$$C = \{c_1, c_2, c_3 \ldots c_n\}$$

where c_k is the color of pixel at (i,j) of the Image I_k from the input image set.

Step 3: Get 'outputColor' by performing required statistical operation on *C*.

$$\text{outputColor} = f(C)$$

where $f(C)$ is a function that takes input a set of colors and outputs a single color. It represents the statistical method taken under consideration.
It can be one of the functions: 'getMean(*C*)', 'getMode(*C*)', 'getMedian(*C*)' or 'getJenksMean(*C*)'

Step 4: Set the color of $P_{i,j}$ of 'outputImage' to outputColor.

setPixel(outputImage,*i*,*j*,outputColor)

The working of the statistical operations getMean(), getMode(), getMedian(), getJenksMean() will be discussed in the next section.

The respective algorithmic complexities of the mean, mode and median methods are $O(w*h*n)$, $O(w*h*n^2)$ and $O(w*h*n*\log n)$ where '*w*' is the width of the images, '*h*' is the height of the images and *n* is the number of images.

2.4 PERFORMING STATISTICAL OPERATIONS ON PIXELS

Pixels are the most basic building block of all images. They have data containing their location in the form of coordinates and information about their color, transparency, etc.

2.4.1 Colors to Magnitudes:

As color is represented in color models like RGB, HSV, HSI it is a 3-dimensional entity [8]. Statistical operations of Mean, Mode or Median cannot be directly performed on multidimensional data. Therefore it has to be first converted into 1-dimensional entity. This can be achieved through multiple ways, two of which are

- Euclidean distance/Magnitude

$$v = \sqrt{r^2 + g^2 + b^2}$$

- Value Average/Intensity

$$v = (r + g + b)/3$$

From the study of these two techniques in relation to their use in the above algorithm, it is found that the Magnitude works better than the Intensity such that it generates better output across all sample image sets tested.

2.4.2 Mean, Mode, Median:

Once all the colors are converted into uni-dimensional entities, their values can be compared with each other to get the mean, mode or median. Consider the following example. Here we have a set of input colors as follows

RGB (255,133,133)	RGB (6,219,9)	RGB (14,220,0)	RGB (15,200,13)	RGB (5,222,4)

Fig.2. Example input colors for pixel operations explanation

From the above mentioned formula for magnitude calculation, we get the following values for their respective colors.

316	219	220	219	222

Fig.3. Magnitude calculations of input colors shown in Fig.1

The Mean, Mode and Median operations are elementary statistical operations. [7]. Mean can be directly applied to the colors by averaging the RGB channels individually. The other two operations, Mode and Median can be performed on the one dimensional values we calculated just now. The results obtained can be matched to their original colors to get the output color.

Mean	Mode	Median
RGB (55,175,10)	219	220

Fig.4. Results of statistical operations on input colors shown in Fig.1

2.4.3 The Jenks Mean Method:

The Mean operation displays the least accuracy of all three statistical operations. This is because, of 'n' given options to color a particular pixel, corresponding to the particular images of the input dataset, the output color is the average of all the colors. Taking the average means that every element has an equal contribution (of $1/n$) in the final result, even the outliers. But it makes no sense to consider the outliers in the calculation of the output. The outliers, essentially are exceptions to the rule and should not be statistically significant while calculating the result. Favorable results can never be achieved if the exceptions contribute to the final result, however small a contribution it may be. Therefore, the first step would be to get rid of the outliers from our consideration and then perform analysis on the remaining clean data. Identification of the outliers can be done on the fact that irregularities in data are infrequent.

Clustering divides input data according to their relative closeness. As already stated, irregularities in data are infrequent. Therefore the largest cluster formed must contain the values that conform to the natural state. Based on this idea, 'getJenksMean()' function can be defined which works in the following steps.

- Identify 3 clusters of color magnitudes by use of Jenks Natural Partitioning.
- Identify the cluster with the largest size.
- Take the mean of elements belonging to this cluster only.
- Locate the mean value thus obtained in the original input set.

Jenks Natural Partitioning is a modification of k-means clustering suitable for one-dimensional data. It is more efficient than k-means clustering as it makes use of the fact that 1D data can be sorted [9] - [11].

We chose to partition the data into 3 clusters as the correct color would generally lie in the center of the sorted list. In such a case, the central cluster is most likely to contain that element. But even when that is not the case, Cluster 1 or Cluster 3 would expand in their respective directions to include the correct element in them. These above mentioned steps can be better illustrated with the help of the following diagram. We shall continue with the example from Fig.3.

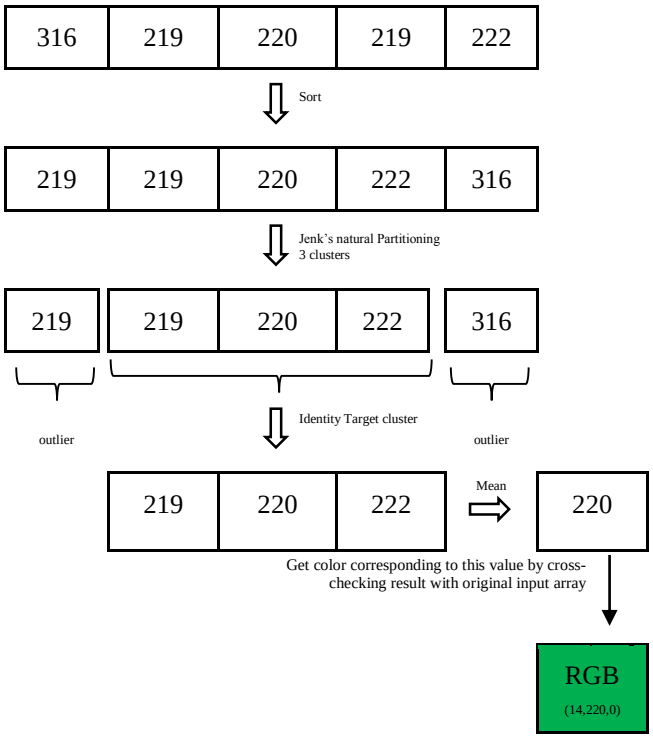


Fig.5. Illustration of Mean Calculation by outlier removal using Jenks Natural Partitioning

The entire process of pixel operations can be summarized in the Fig.6.

3. WORKING PRINCIPLE

As stated earlier, the algorithm works on the belief that the 'majority can never be wrong'. To understand this better, let's consider a simple example.

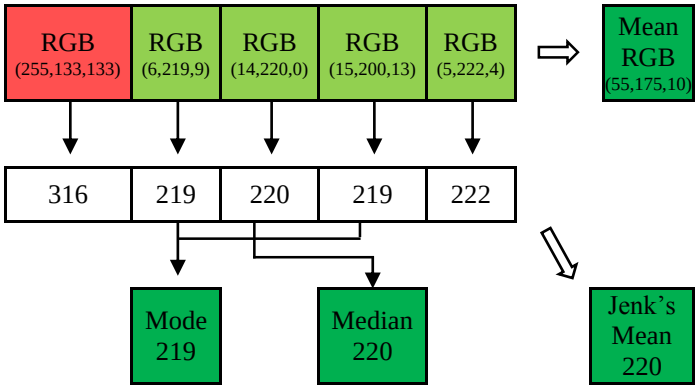


Fig.6. Pixel Operations Overview

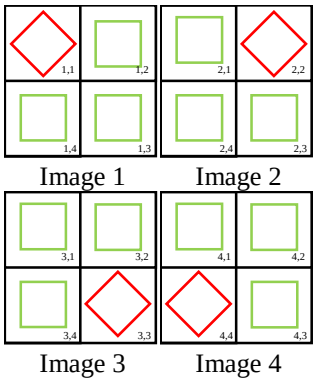


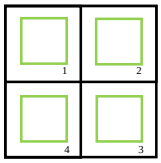
Fig.7. Sample input image set to explain working of the algorithm

Now, each of the above displayed images contains an unwanted object (the red diamond). For easier understanding, the images have been divided into 4 sections. As we can see, the error is contained within the section 1.1, section 2.2, section 3.3 and section 4.4 of images 1, 2, 3 and 4 respectively. Also, the color scheme chosen in these representations makes it clear to the user which part of the image is an error and which is not.

The point to be noted here is that for the computer, the problem isn't subjective, but objective. Appearance of an object may be unwanted for one user, but can be a feature of the image for another user. From a computer's perspective, all that it can see in an image is a collection of pixels. It can take decisions without being biased in any way.

Now the algorithm is expected to take a set of input images, do some processing and generate a single output image. In this example, let us walk through the process of generating the output with regards to individual Sections of the input images. We shall build the output image Section by Section.

For getting the output section 1, we look at all the Section '*x*.1' of input images '*x*'. We are presented with 4 options, 3 of which tell us that it should be occupied with a green square (section 2.1, 3.1 and section 4.1) and one of which tells us that it should be occupied by a red diamond (section 1.1). As mentioned earlier, the algorithm works on the principle of 'Majority is always right'. Therefore, we shall select the green square as the right answer. This same process is repeated for section 2, section 3 and section 4. Finally, we would get an output image that looks like this.

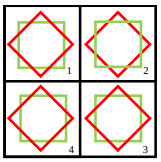


Output Image

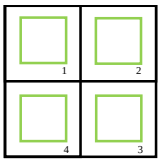
Fig.8. Result of statistical operation (mode) on input images shown in Fig.6

The process discussed above makes use of the 'Mode' operation. Theoretically, the Mode operation provides the best results. This is because the number of entries in the colors set that correspond to the true color outnumber the number of entries corresponding to the unwanted object and therefore should be repeated the most number of times. Consequently, the mode operation would always yield the true color of the pixel taken under consideration. But in reality, for photographs taken with a camera, it is observed that the color of pixels is highly volatile and changes with the slightest change in conditions when the photograph is taken. This results in the scenario that the set of input colors contains all unique values. Therefore calculation of mode becomes impossible.

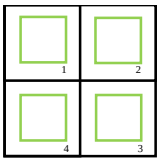
The mean and median operations are pretty straightforward. Median method involves sorting out the colors in order and choosing the color at the midpoint of the list. The color at the midpoint of the list would always be closer to the optimal color because the list would be skewed towards the correct answer. The Mean method simply involves averaging out the colors across all images. The unwanted object isn't completely removed from the images when by performing the mean operation due to the fundamental flaw innate characteristic of the operation that takes equal contribution of all inputs, even the flawed ones. This accuracy is increased by using the Jenks Mean method which provides results of quality comparable to that of the Median method. The final output of all 4 methods is shown below.



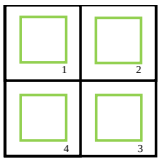
Output image – Mean method



Output image– Mode method



Output image – Median method



Output image – Jenk's Mean Method method

Fig.9. Final output of algorithm on input images

4. SAMPLE IMAGES AND RESULTS

All the artefacts that were covering the main subject have been removed to various degrees of accuracy in the output. It is also worth noting the absence of the minute-hand and hour-hand of the clock in the output images. Even though it may not necessarily be considered as an unwanted object by a human, the algorithm treats it as one and is removed. This further demonstrates the concept that the commonalities between images are retained and the peculiarities are discarded. Each method performs with different accuracy while removing the unwanted characters that appear in the input images. For this sample set mean method works with the least accuracy while the clustered-mean (using Jenks Natural Partitioning) performs the best. Median works better than Mode. The accuracy of all the methods on this sample set can be ranked from best to worst as follows.

- Jenk's Mean
- Median
- Mode
- Mean

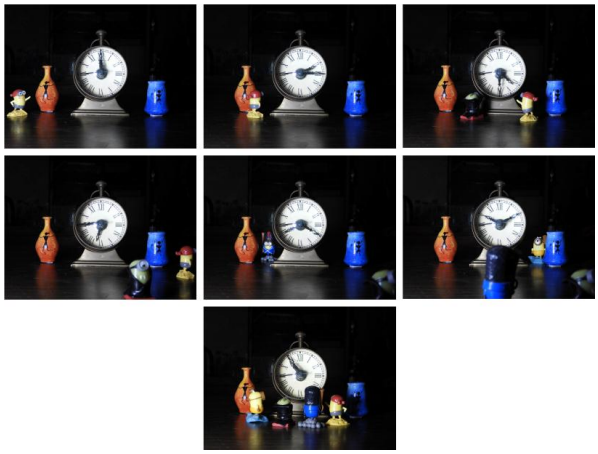


Fig.10. Set of 7 Input Images

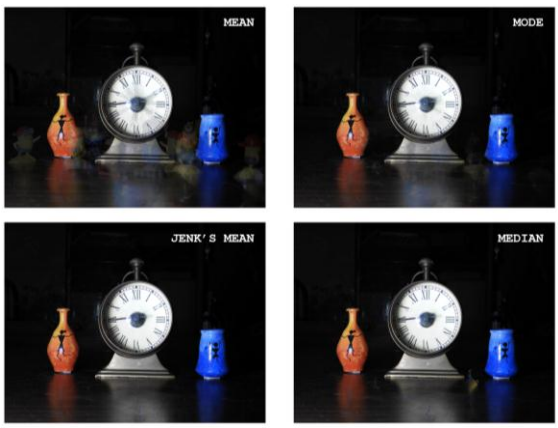


Fig.11. Output Images for all 4 methods

5. RESULTS ANALYSIS AND DISCUSSION

From the results we can infer that Median method works best while Mean method works the worst. Mode method although theoretically sound has some practical limitations which hamper its efficiency.

Table.1. Result Analysis

Sl. No.	Input		Output			
	Quantity	Quality	Mean	Jenks Mean	Mode	Median
1	1	*NA*	O_2	O_2	O_2	O_2
2	2	I_1	O_1	O_1	O_1	O_1
3	2	I_2	O_2	O_2	O_2	O_2
4	2	I_3,I_4	O_{31}	O_2	O_2	O_2
5	3	I_1	O_1	O_1	O_1	O_1
6	3	I_2	O_2	O_2	O_2	O_2
7	3	I_3	O_{31}	O_{32}	O_{32}	O_4
8	3	I_4	O_{32}	O_{31}	O_{32}	O_4
9	>3	I_1	O_1	O_1	O_1	O_1
10	>3	I_2	O_2	O_2	O_2	O_2
11	>3	I_3	O_{32}	O_{33}	O_{33}	O_4
12	>3	I_4	O_{32}	O_4	O_{33}	O_4

Table.2. Index of Labels used in Result Analysis table

Sl. No	Label	Meaning
Input Image Labels		
1	I_1	Subject is moving
2	I_2	Subject stationary, Object stationary
3	I_3	Subject stationary, object moving with overlap
4	I_4	Subject stationary, object moving with no overlap
Output Image Labels		
5	O_1	Subject Distorted
6	O_2	Object not removed
7	O_{31}	Object partially removed with high error
8	O_{32}	Object partially removed with medium error
9	O_{33}	Object partially removed with low error
10	O_4	Object Completely Removed

The mean method performs with the least accuracy while the Jenks Mean method and the Median method perform the best. The accuracy of the Mode is generally lower than these two but more than the Mean method. It should also be noted that the accuracy of all algorithms increases with the increase in the size of input image set.

6. VARIABLES INVOLVED IN THE WORKING OF THE ALGORITHM

6.1 THE NATURE OF THE IMAGES USED

The efficiency of this algorithm is highly dependent on the input set both in terms of quality and quantity of images provided. Generally, the algorithm works better as the size of input set increases. But not strictly. This is because of two things, First, The accuracy of the algorithm also depends on the type of input-how much does the unwanted object move around from image to image, how much position on each of the images is affected by the presence of the unwanted object, how much do the other parameters like brightness, light intensity, etc. change from image to image, how still is the subject and so on. Secondly, beyond a certain point, increasing the input image size has no positive impact on the working of the algorithm. Rather, it may cause adverse effects in terms of not being able to determine the correct color for the pixel, or performing additional unnecessary comparisons, unnecessary iterations even though optimal results have already been obtained.

6.2 STATISTICAL METHOD CHOSEN

As discussed earlier, not all statistical methods perform equally. So the quality output would change vastly depending on the statistical operation chosen.

6.3 STEPS TAKEN WHEN MODE CANNOT BE CALCULATED

Every so often, it happens that for a particular location in the pixel space, all the pixels at that location in different input images have different colors. Therefore, as every value is unique mode cannot be directly calculated based on 'maximum occurrence of a particular value' in this scenario any of the following approaches can be taken.

6.3.1 Using the Empirical Relation between Mean Mode and Median:

The Empirical relation between Mean, Mode and Median is:

$$3(Mean - Median) = Mean - Mode$$

The mode obtained using this relation can be fitted to the colors from the input set matching closest with it.

6.3.2 Taking Mean or Median of the Colors:

This is the hybridization of two statistical operations. First mode is calculated, and only for the exceptions when it cannot be directly calculated, other operation (mean or median) is applied in its place.

6.3.3 Kernels and Convolution Matrices:

Convolution matrices in image processing can be used for all kinds of operations like blurring, sharpening, edge detection, etc. [8]. This process is also known as image filtering. We can also use convolution matrices for high pass filters, low pass filters, unsharp masks, etc. For our use, we can use any of the above kernels individually or in combination to get better results depending on the use case.

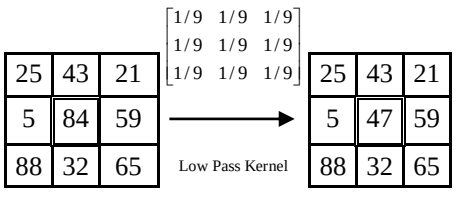


Fig.12. Example of Kernels using a low pass filter

7. ADVANTAGES OF THIS METHOD

7.1 THIS METHOD IS GENERALIZED

There are other methods to remove unwanted objects from images during post-processing, but there are too use-specific. Manual correction and in-painting, both require user to identify or highlight the area containing the discrepancy. In-painting requires a shape mask over the image which determines the correction scope of the algorithm. But, the methods discussed in this paper are generalized. No user involvement is necessary. The computer, on its own decides which objects should be retained and which objects should be discarded based on the statistical analysis.

7.2 NO EXTRAPOLATION

Both in-painting and manual photo correction are based on extrapolation. The accuracy of manual photo correction depends entirely on expertize of the user, while the accuracy of computerized in-painting algorithms depends on the computer's capability to perform educated guesses from the information available in the image (e.g. Neighboring pixels, edge detection, and other data of the image). As opposed to this, the methods discussed here involve no extrapolation. The data and information which is obscured in one image due to the unwanted artifact is extracted from other similar images.

7.3 IMAGE SMOOTHENING AND NOISE REDUCTION

As a positive consequence of the statistical operations, small discrepancies in the image, like noise, get removed during the correction process. This makes the output image smoother that any of the input images.

7.4 MOTION BLUR

Things don't always remain stationary when we take multiple images. For example, grass, tree leaves, etc. are moving constantly as the wind blows. Stacking multiple images with these differences in them can simulate the effect of motion blur. Although generally this is an undesired effect, as it decreases the sharpness of the image and detail are lost, in some cases it can be rather beneficial. If the subject is perfectly stationary, it is in no way affected by this while the background or foreground objects are blurred. This highlights and emphasizes the subject while removing other distractions.

7.5 LESS TIME CONSUMPTION

As compared to the manual work and in-painting, the total time required in the post-processing stage is a lot less in this method.

8. LIMITATIONS AND DISADVANTAGES

8.1 THE NATURE OF THE IMAGES

Getting the right input image set is crucial for this algorithm to work effectively. In some cases one may casually take a series of images from one's smartphone and apply this algorithm to it, and get fabulous results. But in some cases, one may be required to use a tripod mount and a camera with manual mode. The algorithm requires the images to satisfy certain criteria, which may not always be possible.

8.2 NOT 100% ACCURATE

The algorithm may work in most cases that satisfy the input conditions. But even then, in terms of accuracy they can never rival a perfectly clear image taken naturally or error-corrected manually by human expert. We as humans have access to a lot more knowledge about the world and its working than the computer, which just makes use of the information provided in the input set. Therefore, the expert can take better decisions while reconstructing the image after removal of errors

8.3 WORKING ON MAGNITUDES

Converting 3-dimensional color data into magnitude introduces consequent data loss. For example, consider converting Red (RGB-255, 0, 0) and Green (RGB-0, 255, 0) into their respective magnitudes (which is 255). Applying any operations on the magnitudes of these colors will yield the same result. Therefore it becomes difficult to differentiate between input colors solely based on the magnitude.

9. SCOPE FOR IMPROVEMENTS

9.1 IMAGE PREPROCESSING

One of the causes for the reduction of accuracy of the algorithm is that the position of the subject itself changes from image to image. This problem can be solved by applying an image stack aligning algorithm that applies certain transformations to each image in the stack so as to lock the position of a particular object across all images.

9.2 IMAGE POSTERIZATION

Image Posterization is the process reducing the color gamut of the image [8]. This process can be helpful to this algorithm in addressing the problem of mode calculation. With posterization, colors close to each other get closer while the colors dissimilar to each other become more so. This increases the chance of getting repeated colors across pixels and hence enable us to calculate mode.

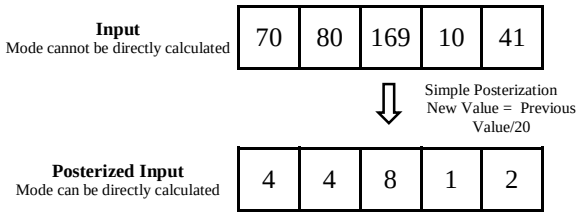
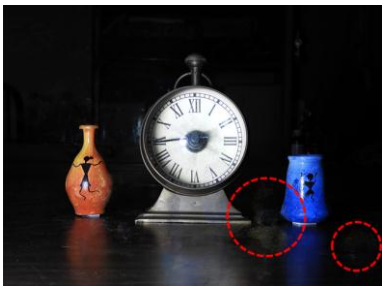


Fig.13. Example of Posterization. The input values ranging from 0-255 are posterized by reducing range 0-13 by simple integer division by 20 and rounding operation



(a) No posterization. Unwanted objects not completely removed

(b) Posterization with Threshold = 25. Unwanted objects completely removed.

(c) Posterization with Threshold = 100. Too much posterization introduces new errors in the output image

Fig. 14. Effects of posterization on quality of output for Mode operation.

9.3 WORKING ON INDIVIDUAL COLOR CHANNELS

The problem encountered with working on the magnitudes of colors can be, solved by applying the same algorithms on individual color channels of pixels rather that the magnitudes of the colors. However, this further increases the complexity and run-time of the algorithm.

9.4 K-MEANS CLUSTERING

Jenks Natural Partitioning was chosen because it is the best method to cluster 1-Dimensional Data. But as we discussed in Section 8.3, information is lost while converting from 3D (color) to 1D (magnitude of color). K-Means clustering can work directly on n-dimensional data. We can use this method for grouping colors.

10. CONCLUSIONS

The methods described in this paper are simple yet highly effective in removing unwanted objects from images, out-performing image in-painting and manual photo correction in terms of accurately reconstructing the image and time and effort required. However it is evident that the nature of images required as input may be hard to obtain and the statistical techniques themselves possess some inherent limitations with respect to this application. There is a tradeoff between the speed and simplicity of the technique and the application scope limited by the nature of input images. Therefore, it is recommended to use this technique in combination with in-painting or manual editing so as to provide the best results.

REFERENCES

[1] S. Ravi, P. Pasupathi, S. Muhukumar and N. Krishnan "Image In-Painting Techniques-A Survey and Analysis", *Proceedings of 9th International Conference on Innovations in Information Technology*, pp. 1-6, 2013.

[2] V.V. Nabiyev, A. Tasci and M. Ulutas, "Removing Unwanted Objects from an Image", *Proceedings of 19th International Conference on Signal Processing and Communications Applications*, pp. 1-7, 2011.

[3] L.B. Bangaru and V. Gupta, "Object Removal by Kriging Interpolation Technique", *Proceedings of International Conference on Cognitive Computing and Information Processing*, pp. 1-4, 2015.

[4] A. Criminisi, P. Prez and K. Toyama, "Object Removal by Exemplar-Based Inpainting", *Proceedings of International Conference on Computer Vision and Pattern Recognition*, pp. 721-728, 2003.

[5] M. Bertalmio, G. Sapiro, V. Caselles and C. Ballester, "Image Inpainting", *Proceedings of International Conference on Graphics and Interactive Techniques*, pp. 417-424, 2000.

[6] A. Criminisi, P. Perez and K. Toyama, "Region Filling and Object Removal by Exemplar-Based Image Inpainting", *IEEE Transactions on Image Processing*, Vol. 13, No. 9, pp. 233-246, 2004.

[7] R. Witte, "*Statistics*", 10th Edition, Wiley, 2017.

[8] Rafael C. Gonzalez and Richard E. Woods, "*Digital Image Processing*", 2nd Edition, Pearson, 2007.

[9] B.D. Dent, "*Cartography*", McGraw-Hill, 2007.

[10] G.F. Jenks, "Generalization in Statistical Mapping", *Annals of the Association of American Geographers*, Vol. 53, No. 1, pp. 15-26, 1963.

[11] G.F. Jenks, "The Data Model Concept in Statistical Mapping", *International Yearbook of Cartography*, Vol. 7, pp. 186-190, 1967.