International Journal of Computer Theory and Engineering, Vol. 2, No. 4, August, 2010
1793-8201

Performance Analysis of Kekre's Median Fast Search, Kekre's Centroid Fast Search and Exhaustive Search Used for Colouring a Greyscale Image

Dr. H. B. Kekre*, Sudeep D. Thepade# and Adib Parkar^

Abstract – **While there exist various techniques that can be used to colour a greyscale image, such as those described in [1], [2], [3] and [4], in this paper we focus on one such technique that can be used with various colour spaces such as RGB, LUV, YCgCb, YCbCr, YUV, XYZ, and YIQ. This technique is described in detail in [4]. In this paper we analyse the performance of Kekre's Median Fast Search, Kekre's Centroid Fast Search and Exhaustive Search algorithms in the greyscale colouring process with respect to time taken for colourization and quality of the coloured image.**

Index Terms—**colour transfer, colour palette, colour spaces, pixel windows, colourization.**

I. Introduction

While colour images are now quite commonplace, there was a time when all images (and videos) were solely in greyscale due to limitations in technology. For many of these images, it is highly desirable to have a coloured version to provide a greater touch of realism. Similarly, there is also interest in being able to convert old feature films that are entirely in greyscale to an acceptable coloured version.

Many techniques have been proposed to achieve this effect such as those described in [1], [2] and [3]. However, these techniques have the inherent drawback of needing a certain amount of human intervention as the colourization process is being applied, such as choosing a seed pixel, assigning it a colour and so on. In this paper, however, we concentrate on the technique described in [4], which requires minimal human interaction. All that is needed is a source image of a similar scene to the target image being constructed. Thus, the technique outlined in [4] can be automated, allowing for the efficient generation of coloured images from given greyscale images provided coloured images of similar scenes are given.

The greyscale image colourization technique described in [4] works on the principle of mapping greyscale (or luminance) values to corresponding colour space values that can be used to reconstruct the original colour. Since there exists a many to one mapping between colour values and their corresponding luminance values, if a pixel-by-pixel mapping is constructed, the probability of finding the correct match for a given luminance value is extremely low. Thus, to improve the probability of finding a correct (or near correct) match, more than one pixels are taken at a time while constructing the luminance-to-colour (or gray-to-colour) mapping table.

In [4], four pixels were grouped together to form a pixel window grid size of 2 x 2 pixels. Thus each entry in the mapping table corresponded to four pixels in the image. In this paper, the grid size is varied from a grid of 1 x 2 pixels to 3 x 3 pixels, along with 4 x 1, 4 x 2 and 2 x 4 pixel grids. The largest grid size constructed in terms of the number of pixels involved is a grid size of 3 x 3 pixels (that is, nine pixels).

The main aim of this paper, however, is to compare the performance of different search algorithms that can be used to find a match in the luminance-to-colour (or gray-to-colour) mapping table. The search algorithms that are used in this paper are the Exhaustive Search (ES) algorithm, Kekre's Median Fast Search (KMFS) algorithm, and Kekre's Centroid Fast Search (KCFS) algorithm. The KMFS algorithm has also been used in [4] but was identified as Kekre's Fast Search algorithm.

Each of these search algorithms is meticulously studied in this paper, and a comprehensive comparison is obtained by testing the search algorithms over various images and also by varying pixel grid sizes. Thus, this paper attempts to give a comprehensive analysis of the performance of the search algorithms that can be used for colouring a greyscale image using the procedure delineated in [4]. Performance of the search algorithms is measured in terms of time taken for a single search, time taken for colouring an entire greyscale image and the quality of the coloured image. When the source (colour) and target (greyscale) images used are the same, quality is measured in terms of the Mean Squared Error (MSE).

II. Colour Spaces Used for Experimentation

A. RGB Colour Space (standard)

The RGB colour space is the standard red-green-blue colour space used when constructing a colour image. The R, G and B values indicate the red, green and blue components respectively of the colour of the pixel. The R, G and B values can vary from 0 to 255, thus allowing for the

*Senior Professor, #Assistant Professor and PhD Research Scholar, ^B. E. Computers Student
*,#Mukesh Patel School of Technology, Management and Engineering, SVKM's NMIMS University, Vile Parle (W), Mumbai – 56
^Thadomal Shahani Engineering College, Bandra (W), Mumbai – 50, India.
(Email:*hbkekre@yahoo.com,#sudeepthepade@gmail.com,^adibparkar @gmail.com)

construction of 24 bit colour images. The luminance is calculated using a weighted average of the R, G and B values such that the sum of the weights is unity.

B. Kekre's LUV Colour Space

Kekre's LUV colour space is introduced in [5], [6] and [7], and is related to the standard RGB colour space as follows:

$$\begin{bmatrix} L \\ U \\ V \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -2 & 1 & 1 \\ 0 & -1 & 1 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & -2 & 0 \\ 1 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix} . \begin{bmatrix} L/3 \\ U/6 \\ V/2 \end{bmatrix} \quad (2)$$

The L/3 component provides the luminance values. The Kekre's LUV color space is special case of Kekre's Transform [14],[17].

C. Kekre's YCgCb Colour Space

Kekre's YCgCb colour space is a newly introduced colour space and is related to the standard RGB colour space as follows:

$$\begin{bmatrix} Y \\ Cg \\ Cb \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{3} . \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix} . \begin{bmatrix} Y \\ Cg \\ Cb \end{bmatrix} \quad (4)$$

The Y/3 component provides the luminance values.

D. YCbCr Colour Space

The YCbCr colour space is currently used extensively and its use in similar applications are researched in [8] and [9]. It's relation with the RGB colour space is as follows:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.2989 & 0.5866 & 0.1145 \\ -0.1688 & -0.3312 & 0.5 \\ 0.5 & -0.4184 & -0.0816 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (5)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & -0.001 & 1.402 \\ 1 & -0.3441 & -0.714 \\ 1 & 1.7718 & 0.001 \end{bmatrix} . \begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} \quad (6)$$

The Y component provides the luminance values.

E. YUV Colour Space

The YUV colour space is used in the PAL (Phase Alternation Line), NTSC (National Television System Committee) and SECAM (Sequentiel Couleur Avec Mémoire or Sequential Colour with Memory) composite colour video standards. It's effectiveness in other applications is studied in [10] and [11]. The following matrices may be used to interconvert between the YUV and RGB colour spaces:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51498 & 0.10001 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (7)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0.74952 & -0.50901 & 1.1398 \\ 1.0836 & -0.22472 & -0.5806 \\ 0.97086 & 1.9729 & 0.00001467 \end{bmatrix} . \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \quad (8)$$

The Y component (as in the YCbCr colour space) provides the luminance values.

F. XYZ Colour Space

The XYZ colour space is not as widely used as the YCbCr, YUV and YIQ colour spaces. Its relationship with the RGB colour space is given as follows:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412456 & 0.3575761 & 0.1804375 \\ 0.212673 & 0.7151522 & 0.0721750 \\ 0.019334 & 0.1191920 & 0.9503041 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (9)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.240454 & -1.53714 & -0.49853 \\ -0.96927 & 1.876011 & 0.041556 \\ 0.055643 & -0.20403 & 1.057225 \end{bmatrix} . \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (10)$$

G. YIQ Colour Space

The YIQ colour space is derived from YUV colour space and is optionally used by the NTSC composite colour video standard. Its transformations are researched in [12]. The *I* stands for in phase and *Q* for quadrature, which is the modulation method used to transmit the colour information. The YIQ and RGB interconversion matrices are as follows:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} . \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (11)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.272 & -0.647 \\ 1 & -1.107 & 1.704 \end{bmatrix} . \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \quad (12)$$

As in all the above Y based colour spaces, the Y component provides the luminance values.

III. Colour Transfer Algorithm

The colour transfer algorithm is discussed in detail in [4] specifically for the LUV colour space with a 2 x 2 pixel grid size. The main steps of the algorithm for a general colour space with an m x n pixel grid size are:

A. Build luminance-to-colour mapping table

The luminance-to-colour mapping table (or gray-to-colour mapping table) is constructed as explained previously in the Introduction using a provided colour image of preferably a similar scene to the greyscale image which is to be coloured. The more similar the scene, the better are the results that are obtained.

B. Process the greyscale image one pixel grid at a time

Read one m x n grid of pixels from the greyscale image that is to be recoloured and find the closest match of the luminance values in the mapping table. The closest match is determined by calculating the Euclidean distance between the luminance values in the mapping table and the luminance values of the pixel grid from the greyscale image. Assuming there are g = m x n pixels in the grid, the Euclidean distance *d* would be calculated for each grid from the equation 13.

$$d^2 = \sum_{i=1}^{g} \left(luminance(i)_{table} - luminance(i)_{image} \right)^2 \quad (13)$$

The mapping table entry with the smallest value of *d* must be found. It must be noted that this does not guarantee the best match, but with a moderate grid size, errors can be

minimized while still producing acceptable results with little or no distortion.

This paper focuses on this step of the colour transfer algorithm by comparing different techniques to find the best match in the mapping table.

C. Recolour the greyscale image

The greyscale image is recoloured using the best match found in step two of the algorithm one pixel grid at a time.

IV. Exhaustive Search (ES)

A. The Algorithm

Exhaustive search is a very simple algorithm to implement, but its performance deteriorates as the image sizes increase or pixel grid sizes decrease.

In exhaustive search, every luminance entry in the luminance-to-colour mapping table is compared with the luminance values obtained from the greyscale image and the Euclidean distance is calculated for each. This is extremely expensive in terms of computational power as the size of the mapping table increases because Euclidean distance calculations are not trivial additions but comprise complex operations such as summations and squares.

Only when the entire table is searched and the minimum Euclidean distance is found, this algorithm terminates and returns the entry with the minimum Euclidean distance.

B. Time Complexity

Consider an image of size M x N pixels and a grid size of m x n pixels with m < M and n < N. Assume that M modulo m = 0 and N modulo n = 0. This assumption ensures that the image is divided into an integral number of grids. This is a reasonable assumption to make, as any image can be padded or resized to fulfil the above conditions.
Let,

$$k = \frac{M}{m} \times \frac{N}{n} \tag{14}$$

Thus the number of entries in the mapping table will be *k*. As the entire table must be searched, the complexity of this algorithm using the standard big-O notation is O(k). Thus the time complexity of exhaustive search increases linearly with increase in either of the dimensions (width or height) of the image.

V. Kekre's Median Fast Search (KMFS)

Kekre's Median Fast Search algorithm was first used in [4] for a grid size of 2 x 2 pixels and was identified as Kekre's Fast Search algorithm. It is now extended to a grid size of m x n pixels to perform a thorough comparison with the exhaustive search algorithm both in terms of quality of output as well as in terms of speed of computation. Note that if the source colour image and the target greyscale image (which is to be coloured using the colour palette generated using the source colour image) are different, and the actual coloured version of the greyscale image is not available, the "quality" of the output is subjective.

A. The Algorithm

Kekre's Median Fast Search algorithm, to a certain degree, is similar to the standard binary search algorithm that is extensively used for its low time complexity. Let g = m x n be the number of pixels found in each pixel grid. Thus every mapping table entry will have g luminance values. Let the values be A_1 to A_g.

First sort the *entire* mapping table with respect to A_1 in either ascending or descending order (either will do). We chose to sort the table in ascending order.

Save the value of A_1 found at the midpoint of the table after sorting. Split the table into two equal halves (or as near as possible into two equal halves) at this position (include the midpoint into the upper half).

Now sort the entire upper half of the table and the entire lower half of the table independently of each other with respect to A_2. Note that the values in A_1 will now no longer be sorted as we are now sorting the table values with respect to A_2. Now save the two values of A_2 found at the respective midpoints and once again split the table in the same manner (thus now there will be four parts).

Repeat the process till sorting with respect to A_g, saving all the midpoints. Once the table entries are sorted as such, the table building process is said to be completed.

When searching the table, first compare the A_1 luminance value of the greyscale image pixel grid with the A_1 value of the first midpoint that had been saved. If the image luminance value is less than or equal to the midpoint value, it is very likely that the best match for this pixel grid will be in the upper half of the table (assuming we have sorted in ascending order). Now compare the value A_2 of the image pixel grid with the A_2 value of the upper midpoint of the second lot of midpoints. Repeat the process till we reach the comparison of the A_g value with the one of the values in the g^{th} set of midpoints.

Now we will have a small set of table entries that are likely to contain the correct mapping entry. The Euclidean distance needs to be calculated only for this small set of entries.

The following figure (Figure 1) illustrates how a search using the KMFS algorithm would proceed for a 4 pixel grid size:

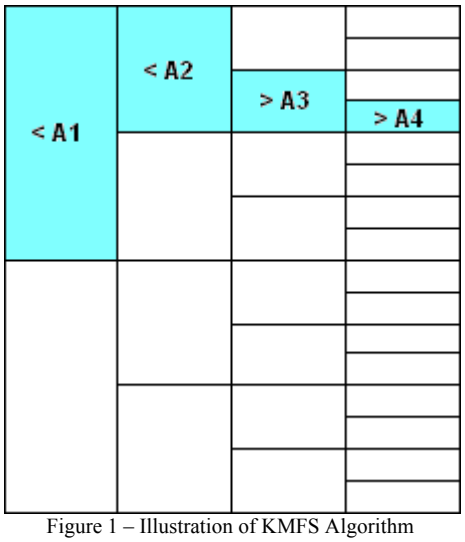


Figure 1 – Illustration of KMFS Algorithm

Only the entries in the coloured block of the final column in the mapping table need to have their Euclidean distances calculated with the greyscale image pixel grid luminance values to find the best match (that is, the entry with the minimum distance). Clearly the number of computations is reduced drastically when the KMFS algorithm is used as compared to the exhaustive search algorithm.

However, it must be noted that now there is a possibility that the table entry with the actual minimum Euclidean distance is not found due to an incorrect branch being chosen. This scenario may occur because the selection of a branch of the table is based on only one of the luminance values in the pixel grid and not all. The probability of such an occurrence increases with increase in grid size.

B. Time Complexity

Consider an image of size M x N pixels and a grid size of m x n pixels with m < M and n < N.

Assume that M modulo m = 0 and N modulo n = 0. This assumption ensures that the image is divided into an integral number of grids. This is a reasonable assumption to make, as any image can be padded or resized to fulfil the above conditions.

Let,

$$k = \frac{M}{m} \times \frac{N}{n} \tag{15}$$

Thus the number of entries in the mapping table will be *k*.

Let g be the grid size,

$$g = m \times n \tag{16}$$

Searching now comprises of g simple comparisons followed by the calculation of approximately $k/2^g$ Euclidean distances.

From (15) and (16) we find,

$$g = \frac{MN}{k} \tag{17}$$

Thus, the complexity of the KMFS algorithm in the standard big-O notation is:

$$O\left(k/2^{\frac{MN}{k}}\right)$$

Hence, Kekre's Median Fast Search will always outperform exhaustive search with regard to time complexity.

Kekre's Centroid Fast Search (KCFS)

Kekre's Centroid Fast Search algorithm [15],[16] is a variation of Kekre's Median Fast Search algorithm.

The Algorithm

Let g = m x n be the number of pixels found in each pixel grid. Thus every mapping table entry will have g luminance values. Let the values be A1 to Ag.

Now instead of finding the median of the values in A1 (i.e. sorting the values and finding the middle value), compute the mean of the values in A1. Now split the table into two parts with all values of A1 that are less than the mean in one section and all values that are greater than the mean in the other section. Note that just as in the case of Kekre's Median Fast Search, when arranging luminance values with respect to A1, all the other luminance values get arranged accordingly as well.

Once the table is split with respect to A1, consider the two sections separately and split them with respect to A2 using the same procedure. Repeat the process till Ag is reached. It must be noticed that while in the KMFS algorithm, the table is split into almost equal sections, when using the KCFS algorithm, the sections obtained need not necessarily be equal.

The search procedure is almost identical to the process described in the KMFS algorithm, except that in this case each luminance value is compared with the mean value rather than the midpoint (median) value. If it is less than the mean, the first section is chosen for a search for the next luminance value, else the second section is chosen. This continues till a comparison has been made at the Ag level. However, the reduction in the number of Euclidean distance calculations needed using this search algorithm may not be as great as the reduction obtained using the KMFS algorithm due to the possibility of skewed section sizes.

In addition, the drawback of the KMFS algorithm also applies to the KCFS algorithm. It is possible for the KCFS algorithm also to choose an incorrect branch leading to the best possible match for the luminance values to not be found.

Time Complexity

Consider an image of size M x N pixels and a grid size of m x n pixels with m < M and n < N.

Once again, as in the case of Kekre's Median Fast Search, assume that M modulo m = 0 and N modulo n = 0.

Let,

$$k = \frac{M}{m} \times \frac{N}{n} \tag{18}$$

Thus the number of entries in the mapping table will be *k*.

Let g be the grid size,

$$g = m \times n \tag{19}$$

Thus, there will be g luminance values for each entry in the mapping table (A_1 to A_g).

As in the case of Kekre's Median Fast Search, searching now comprises of g simple comparisons followed by the calculation of Euclidean distances. However, in the case of this algorithm, the number of entries in the mapping table for which the Euclidean distance must be calculated can no longer be approximated to $k/2^g$, as the mapping table is no longer split into equal (or even almost equal) parts.

Let p be the number of divisions in the mapping table for the last entry A_g. Thus, the expected number of entries for which the Euclidean distance must be calculated is approximately k/p.

Thus, the complexity of the KCFS algorithm in the standard big-O notation is:

$$O\left(k/p\right)$$

Note that in the best case, the table will be split into sections of nearly equal sizes and in this scenario $p = 2^g$, thus reducing the time complexity of KCFS to that of KMFS. In general, however, $p < 2^g$.

Hence, Kekre's Centroid Fast Search will always outperform exhaustive search but, in general, will not perform as well as Kekre's Median Fast Search with respect to time complexity.

VI. Comparison of Reconstructed Images Using Various Pixel Grid Sizes (Same Source and Target)

Various images, each of size 180 x 180 pixels, were used to build the mapping table, and their greyscale equivalents were coloured using the constructed mapping table for various grid sizes with searches performed using all the search algorithms discussed previously.

The following bar chart (Figure 2) shows the comparison of the average mean squared error obtained across all images and all seven colour spaces for various grid sizes using the various search algorithms.

As can be seen from the bar chart, Kekre's Median Fast Search actually outperforms Exhaustive Search for the smaller grid sizes, but its performance rapidly deteriorates as the grid sizes increase and become more rectangular. Note how the square grid size of 3 x 3 pixels performs significantly better using the KMFS algorithm as compared to the rectangular grid sizes of 2 x 4 pixels and 4 x 2 pixels with the same algorithm.

Kekre's Centroid Fast Search performs similarly to Kekre's Median Fast Search for smaller grid sizes but performs significantly better than it for the larger, more rectangular grid sizes, though not as well as Exhaustive Search.

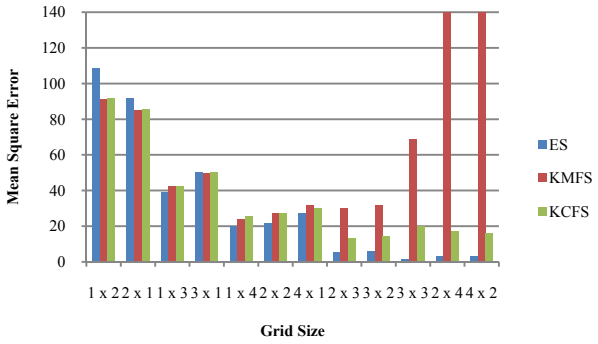


Figure 2 – MSE across various Grid Sizes

The comparison of the search algorithms with respect to time taken per search across the various grid sizes for all images is shown in the following bar chart (Figure 3). The time has been measured in milliseconds.

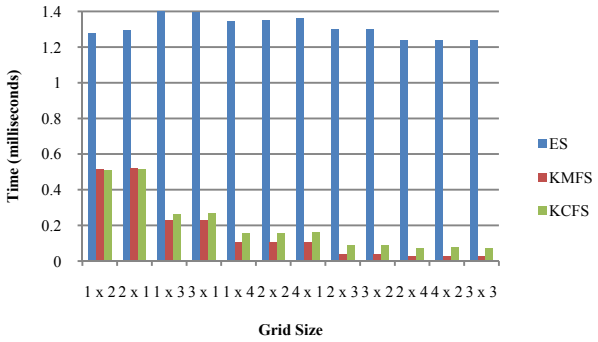


Figure 3 – Time Taken per Search across various Grid Sizes

The comparison of the average time taken to reconstruct an entire image across all images using the search algorithms is shown in Figure 4. Here the time has been measured in seconds. All images, as mentioned previously were of the same dimensions (180 x 180 pixels), thus allowing for a valid comparison of reconstruction times across grid sizes.

Thus, it is actually better to use one of Kekre's Fast Search algorithms whenever a small pixel grid size is being used. They are not only faster, but also produce better results. However, it is also seen that when the grid size is very small, the overall errors are larger as compared to a slightly larger grid size like 2 x 2 pixels in the case of KMFS and 3 x 3 pixels in the case of ES. It is also noticed that the KCFS algorithm thoroughly outperforms the KMFS algorithm in terms of errors for larger grid sizes.

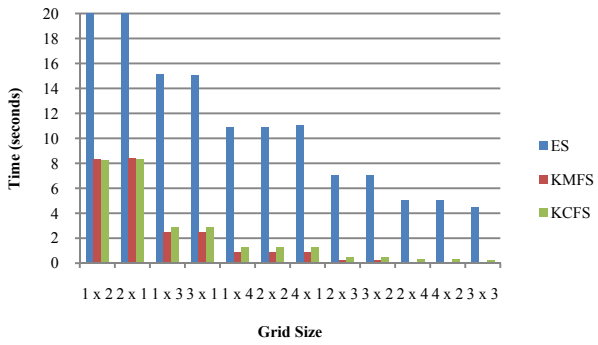


Figure 4 – Time Taken for Reconstruction of Entire Image across various Grid Sizes

When time is considered, Kekre's Fast Search algorithms comprehensively outperform the Exhaustive Search algorithm both in terms of individual search times and in terms of the time taken to reconstruct an entire image. Between the two, the difference is not very great with the KMFS algorithm slightly outperforming the KCFS algorithm.

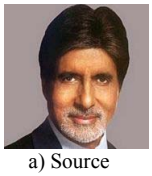
However, when MSE is also taken into account, thanks to the fact that the KCFS algorithm significantly outperforms the KMFS algorithm in this respect for larger grid sizes such as 3 x 3, the KCFS algorithm should be preferred for such grid sizes. For a 2 x 2 grid size or less, the performance of both algorithms is similar with the KMFS algorithm being slightly more efficient with respect to time, so the KMFS algorithm should be preferred.

VII. Comparison of Reconstructed Images Using Various Pixel Grid Sizes (Different Source and Target)

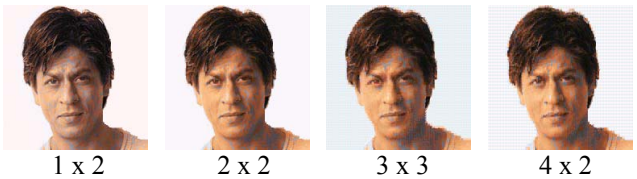
Following are some images that were reconstructed using the search algorithms. Note how both KMFS and ES give comparable results for the smaller grid size but distortions increase for the KMFS algorithm as grid sizes increase. Also note how quality degradation is much lesser for KCFS as compared to KMFS as grid size increases.

As the original colour images of the greyscale images used were not available, the MSE could not be calculated, hence these results are subjective. However, the time taken per search and the time taken for the complete reconstruction of the colour images was measured, and is shown in the bar charts following the images.

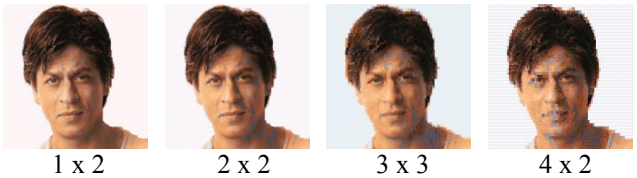
Both the original images (that is the coloured images) and the greyscale images to be coloured were of the same size, 180 x 180 pixels.


a) Source

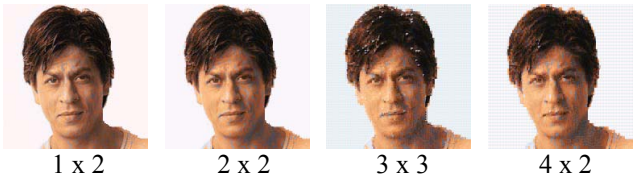

b) Target



c) Exhaustive Search



d) Kekre's Median Fast Search



e) Kekre's Centroid Fast Search

Figure 5 – Reconstruction of target image using similar source image

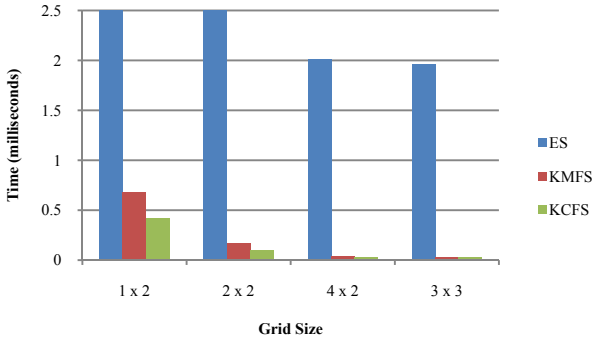


Figure 6 – Time taken per search (in milliseconds)

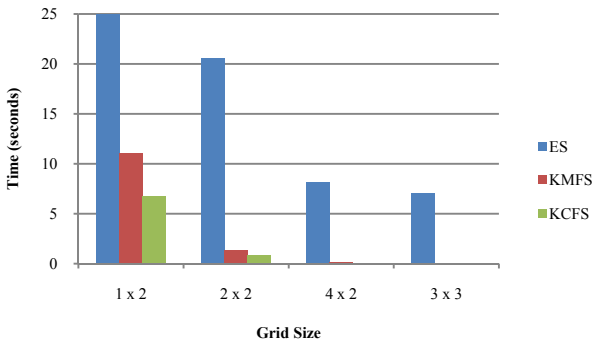
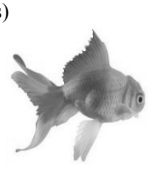
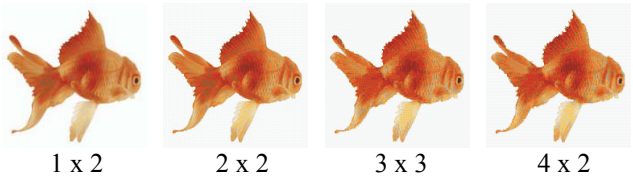


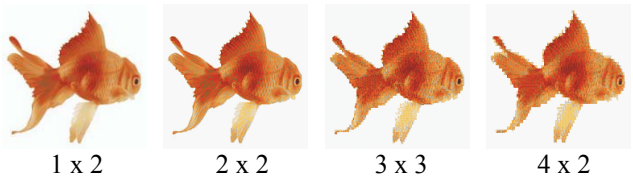
Figure 7 – Time taken for complete reconstruction of the colour image (in seconds)


a) Source

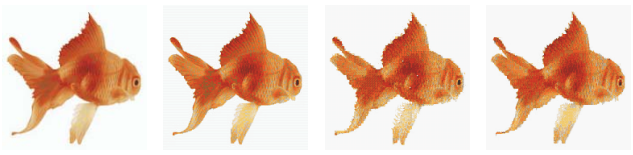

b) Target



c) Exhaustive Search



d) Kekre's Median Fast Search



e) Kekre's Centroid Fast Search

Figure 8 – Reconstruction of target image using similar source image

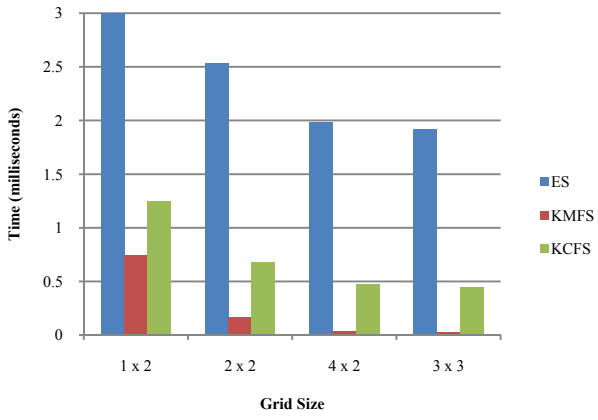


Figure 9 – Time taken per search (in milliseconds)

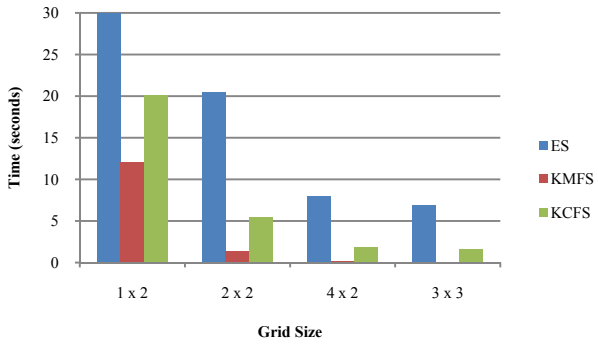


Figure 10 – Time taken for complete reconstruction of the colour image (in seconds)

VIII. CONCLUSION

In this paper, we have done a comprehensive comparison of search algorithms used for colouring greyscale images by the technique described in [4].

We have found that while in general, Kekre's Fast Search algorithms do not perform as well as the Exhaustive Search algorithm with respect to image quality; they still outperform the Exhaustive Search algorithm for smaller pixel grid sizes.

However, we have also proved that Kekre's Fast Search algorithms always perform significantly better than the Exhaustive Search algorithm when time taken is considered. This has been proved both by algorithmic analysis, and by actual measurements.

Between the KMFS and KCFS algorithms we have found that the KMFS algorithm is slightly faster than the KCFS algorithm in general with similar image quality results for smaller pixel grid sizes. On the other hand, as the pixel grid sizes increase above 2 x 2 pixels, the errors obtained using the KMFS algorithm drastically increase. While errors also increase when using the KCFS algorithm, the increase is significantly lesser than that of the KMFS algorithm.

Thus, in conclusion, it can be stated that when using pixel grid sizes with areas up to 4 pixels (that is, 2 x 2 pixels), Kekre's Median Fast Search algorithm should be used due to its highly superior performance in terms of time and acceptable performance in terms of image quality. For larger grid sizes (for example, 3 x 3 pixels) Kekre's Centroid Fast Search algorithm should be preferred instead.

REFERENCES

[1] T. Welsh, M. Ashikhmin, and K. Mueller, Transferring color to grayscale images," ACM TOG, vol. 20, no. 3, pp. 277–280, 2002.
[2] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," IEEE Computer graphics and applications, vol. 21, no. 5, pp. 34–41, September/October 2001.
[3] H. B. Kekre, Sudeep D. Thepade, "Creating the Color Panoramic View using Medley of Grayscale and Color Partial Images", WASET International Journal of Electrical, Computer and System Engineering (IJECSE), Volume 2, No. 3, Summer 2008.
[4] H. B. Kekre, Sudeep D. Thepade, "Color Traits Transfer to Grayscale Images", In Proc. of Int. Conference on Emerging Trends in Engg. And Tech., ICETET-2008.
[5] H. B. Kekre, Sudeep D. Thepade, "Image Blending in Vista Creation using Kekre's LUV Color Space", In SPIT-IEEE Colloquium, SPIT Mumbai, INDIA, Feb 4-5,2008.
[6] H. B. Kekre, Sudeep D. Thepade, "Improving 'Color to Gray and Back' using Kekre's LUV Color Space", IEEE International Advanced Computing Conference 2009 (IACC '09), Thapar University, Patiala, INDIA, 6-7 March 2009.
[7] H. B. Kekre, Sudeep D. Thepade, "Boosting Block Truncation Coding using Kekre's LUV Color Space for Image Retrieval", WASET International Journal of Electrical, Computer and System Engineering (IJECSE), Volume 2, No.3, Summer 2008.
[8] Son Lam Fung, A. Bouzerdoum, D. Chai, "A novel skin color model in YCbCr color space and its application to human face detection", In Proc. of International Conference on Image Processing (ICIP-2002), Vol.1, pp. I289-I292.
[9] Hideki Noda, Michiharu Niimi, "Colorization in YCbCr color space and its application to JPEG images", Pattern Recognition Society Published by Elsevier B.V., Vol.40, number 12, pp.3714-3720, December, 2007.
[10] H. B. Kekre, Sudeep D. Thepade, "Using YUV Color Space to Hoist the Performance of Block Truncation Coding for Image Retrieval", IEEE International Advanced Computing Conference 2009 (IACC '09), Thapar University, Patiala, INDIA, 6-7 March 2009.
[11] Daniela Stanescu, et. al., "Steganography in YUV color space", IEEE International Workshop on Robotic and Sensors Environments, ROSE-2007, Ottawa, Canada, 12-13 October 2007.
[12] B. Ahirwal, M. Khadtare, R. Mehta, "FPGA based system for color space transformation RGB to YIQ and YCbCr", In Proc. of Int. Conference on Intelligent and Advanced Systems, ICIAS 2007, Kuala Lumpur, pp.:1345-1349, 25-28 Nov 2007.
[13] H. B. Kekre, Sudeep D. Thepade, "Color Based Image Retrieval using Amendment Block Truncation Coding with YCbCr Color Space", International Journal on Imaging (IJI), Volume 2, Number A09, Autumn 2009, pp. 2-14.
[14] H. B. Kekre, Sudeep D. Thepade, "Image Retrieval using Non-Involutional Orthogonal Kekre's Transform", International Journal of Multidisciplinary Research and Advances in Engineering (IJMRAE), Ascent Publication House, Volume 1, No.I, 2009.
[15] H.B.kekre, Tanuja K. Sarode, "Centroid Based Fast Search Algorithm for Vector Quantization", International Journal of Imaging (IJI), Volume 1, Number A08, pp. 73-83, Autumn 2008.
[16] H.B.kekre, Tanuja K. Sarode, "Fast Codevector Search Algorithm for 3-D Vector Quantized Codebook", WASET International Journal of Computer and Information Science and Engineering (IJCISE), Volume 2, Number 4, pp. 235-239, Fall 2008.
[17] H. B. Kekre, Sudeep D. Thepade, "Improving the Performance of Image Retrieval using Partial Coefficients of Transformed Image", International Journal of Information Retrieval, Serials Publications, Volume 2, Issue 1, 2009, pp. 72-79.

Dr. H. B. Kekre has received B.E. (Hons.) in Telecomm. Engg. from Jabalpur University in 1958, M.Tech (Industrial Electronics) from IIT Bombay in 1960, M.S.Engg. (Electrical Engg.) from University of Ottawa in 1965 and Ph.D. (System Identification) from IIT Bombay in 1970. He has worked Over 35 years as Faculty of Electrical Engineering and then HOD Computer Science and Engg. at IIT Bombay. For last 13 years worked as a Professor in Department of Computer Engg. at Thadomal Shahani Engineering College, Mumbai. He is currently Senior Professor working with Mukesh Patel School of Technology Management and Engineering, SVKM's NMIMS University, Vile Parle(w), Mumbai, INDIA. He has guided 17 Ph.D.s, 150 M.E./M.Tech Projects and several B.E./B.Tech Projects. His areas of interest are Digital Signal processing and Image Processing. He has more than 250 papers in National / International Conferences / Journals to his credit. Recently six students working under his guidance have received best paper awards. Currently he is guiding ten Ph.D. students.

Sudeep D. Thepade has Received B.E.(Computer) degree from North Maharashtra University with Distinction in 2003. M.E. in Computer Engineering from University of Mumbai in 2008 with Distinction, currently Perusing Ph.D. from SVKM's NMIMS University, Mumbai. He has more than 06 years of experience in teaching and industry. He was Lecturer in Dept. of Information Technology at Thadomal Shahani Engineering College, Bandra(W), Mumbai for nearly 04 years. Currently working as Assistant Professor in Computer Engineering at Mukesh Patel School of Technology Management and Engineering, SVKM's NMIMS University, Vile Parle(W), Mumbai, INDIA. He is member of International Association of Engineers (IAENG) and International Association of Computer Science and Information Technology (IACSIT), Singapore. His areas of interest are Image Processing and Computer Networks. He more than 48 papers in National/International Conferences/Journals to his credit with a Best Paper Award at International Conference SSPCCIN-2008 and Second Best Paper Award at ThinkQuest-2009 National Level paper presentation competition for faculty.

Adib Parkar is currently pursuing a Bachelors (B.E.) degree in Computer Science from Thadomal Shahani Engineering College in Mumbai, India. He has been an active IEEE Student Member for 3 years and is also a member of the Computer Society of India. His areas of interest lie in the fields of Image Processing and Artificial Intelligence.