Sequential Overlap to Find Exact Nearest Patch Match

Manasa.V[#1], Vinay.A[*2]

[#]*M.Tech Student, Department of ISE, PESIT*

**Assistant Professor, Department of ISE, PESIT*

INDIA

Abstract— *Searching set of images to find similar patches in computer vision is a very expensive operation and time consuming The aim of the paper is to provide fast efficient exact nearest patch matching algorithm, in brute-force style. The core matching algorithm - Patch Match, can find similar regions or "patches" of an image. We compare source image with the target image to find exact nearest matches for the patch using sequential overlap concept. Distance between patches is found using Euclidean distance formula. As adjacent patch-pairs overlap to certain extent, redundant calculations are eliminated thereby reducing the time complexity. To further accelerate performance, method is improved by processing multiple columns at a time. Compared to existing methods, proposed method's memory requirement, time complexity is also minimal and does not require auxiliary data structures, so that it can be applied to large image processing applications. As method follows brute-force approach, nearest neighbours or matches found are exact, accurate and optimal. In this work, we compare and evaluate exact nearest neighbour algorithms for speeding up this task.*

Keywords— Nearest patch match, Sequential overlap, Multiple Columns, Patch Match

I. INTRODUCTION

Finding nearest similar patches in images is an expensive step in a wide variety of computer vision tasks. An exhaustive search scans through the entire image for all N patches and take several minutes or even hours. A patch is a small squared area of pixels, i.e. 3x3, 5x5 or 7x7 pixels.

To find nearest patch match in source image Z with m pixels when compared with target image X, brute force implementation compares each pixel of the images and exhibits time complexity of $O(mr^2)$ for patch size of r. Using concept proposed by Huang [8] and Weiss [5], time complexity is reduced from $O(mr^2)$ to $O(mr)$ and redundant calculations can be eliminated. Each image patch is considered as a point in a high-dimensional space and by using concept of sequential overlap that exists among adjacent rows, nearest neighbours or matches for the patches is found in fraction of time. Nearest patch matching depends on factors like image size, search range, number of images to be processed and patch size.

Most of the traditional match matching algorithms like kd-tree [9], TSVQ [4], and ANN [7] find approximate nearest patch and do not use sequential overlap concept between patches. These methods are not optimal and accurate. Brute-force finds exact nearest match for the patch but as image size or patch size increases this method becomes infeasible and takes long processing time.

We solve the following problem: For every patch in source image, we find the exact nearest similar match for the patch in target image using sequential overlap that exists among adjacent rows and columns which solves above problem.

This paper is organized as follows. Section II describes the related work carried in the field of nearest neighbor methods Section III explains how proposed method can be a solution for reducing time complexity and eliminating redundant calculations. Section IV gives the results achieved and some brief discussion on obtained results. Section V concludes the paper and suggests future enhancements.

II. RELATED WORK

Patch matching is used for editing and processing of images in still images or in video. This is done for information retrieval, object recognition, object categorization and object class detection, entire image classification, texture synthesis, object removal, photo reshuffling [2].

Nearest patch search algorithms is roughly classified into two categories: the exact nearest patch matching and approximate nearest patch matching. PCA Trees, K-means, are often used to achieve exact nearest patch matching. Currently traditional used methods, such as kd-Tree, ANN, TSVQ and Vantage Point Trees, can perform both exact and approximate nearest patch matching. These use hierarchical data structures for searching.

Brute-force computation is naive method for finding nearest neighbor of distances between all pairs of points in the dataset: for 'r' number of pixels in patch, in 'm'-number of pixels in image, this approach scales as $O(mr^2)$. Brute-force method is efficient for small data samples. This approach becomes infeasible when number of samples to search, patch size or image size increases.

When sequentially performing nearest patch matching in brute-force style between the source image and the target image, the adjacent patch pairs overlap to certain extent. Using sequential overlaps, redundant calculations and time complexity is reduced.

Processing of multiple columns at a time, presented by Huang [8] the use of sequential overlaps between adjacent windows to eliminate the redundant calculations among columns. Weiss [5] presented multiple columns processing at a time, thereby reducing the time complexity and eliminating redundant calculations among adjacent columns. Sequential overlap consolidates the redundant calculations, reducing the time complexity to O(r).

III. PROPOSED METHOD

1. Sequential Overlap

Patch distance is calculated using standard Euclidean distance function to find the similarity distance between source image and target image by summing the squared (Euclidean) per-pixel distance for each pixel in the given patches between the 2D patch X_i in the target image X and Z in the source image Z_j with patch size 'r' can be computed as:

$$d(X_m, Z_n) = \sum_{i=1}^{r} \sum_{j=1}^{r} [X_m(i,j) - Z_n(i,j)]^2.$$

where (i,j) represents the index position for each pixel inside a patch.

As adjacent patch pairs overlap, redundant calculations are eliminated using sequential overlap in each row, thereby time complexity for 2D patch is reduced from $O(mr^2)$ to O(mr),where 'm' is number of pixels in source image Z.

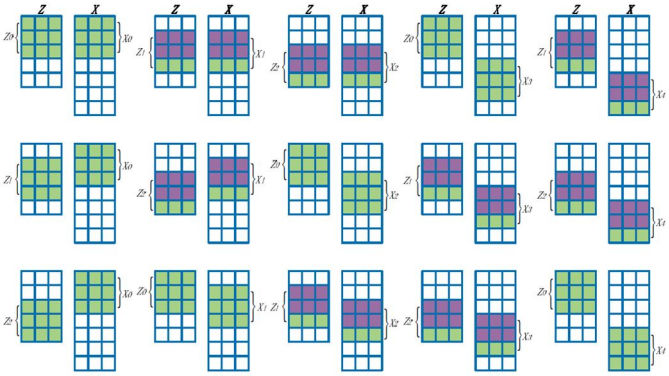

Fig 1: Fast exact nearest patch match using sequential overlap

The height of target image is more than the height of source image. Patches in both source and target image start from the first row: In first iteration L=0, (Z_0, X0); (Z1, X1); (Z2, X2) are simple examples of sliding of patches by one row at a time. (Z_0, X_0) patch pair is initially compared and distance is calculated. Next patch window slides down by one row. Now, (Z_1, X_1) is compared: As adjacent patch pairs overlap, redundant calculations are eliminated by considering similarity distance calculated in previous patch pair and by adding the newly added bottom row distance values. In, (Z_2, X_2) scenario, there is no way to move the source patch down. Therefore it goes back to the top row (Z_0) but patch continues to slide down in target image (X_3) and finally it hits its own last row (X_4).

In the second row in figure, source patch starts from the 2nd row but target patch again starts from the 1st row. Rest of the logic doesn't change. In the 3rd row, the logic repeats. The patch with minimum distance is said to be the exact match for the patch between source and target images. Using sequential overlap, time complexity is reduced to O(mr).

2. Multiple Columns Processing

Performing the nearest patch matching for 'N' columns by comparing 'N' adjacent columns of patches in target image X with 'N' adjacent columns of patches in source image Z, the performance can further be accelerated. Weiss [5] presented processing of multiple columns at a time thereby eliminating redundant calculations. As adjacent columns overlap, processing of columns is done at faster rate compared to processing of single column as discussed previous section.

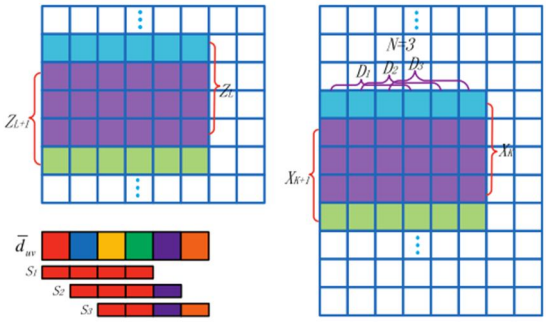

Fig 2: Multiple columns processing

Here patch window is moved from left to right and top to bottom. A bottom row of pixel-pairs (green) are newly added, as shown in figure. The distance between adjacent columns is found. Then the subsequent columns distance is found by eliminating the redundant computations of adjacent columns.

Number of columns to be processed is set as N, where (N = r – 1). If r = 3, then N = 2. Processing of 'N' columns involves computing 'N' dependent patches. As in figure, distance of the newly added row (bottom row of pixels) is found and is added to the previously computed patch pair distance of the top rows, by subtracting with top row values and by subtracting the previous column values. The time for computing distances between each patch-pair is still O(r).

IV. RESULTS

We have implemented above approach using OpenCV 2.1 and Microsoft Visual Studio Professional 2010 in C Language on Intel Core i5 CPU @ 2.67GHz with 4GB RAM.

Below is the table that represents time taken to find patch matches between different size source image and target image.

I. Time Computation in Different Image Sizes in Brute-force and Single Column

Image Size (Source Image, Target Image)	Execution time of Single Column in seconds	Execution time of Brute Force in seconds
80*80	33	204
81*60,90*90	36	196
128*128,150*150	316	2036
160*160	625	3843

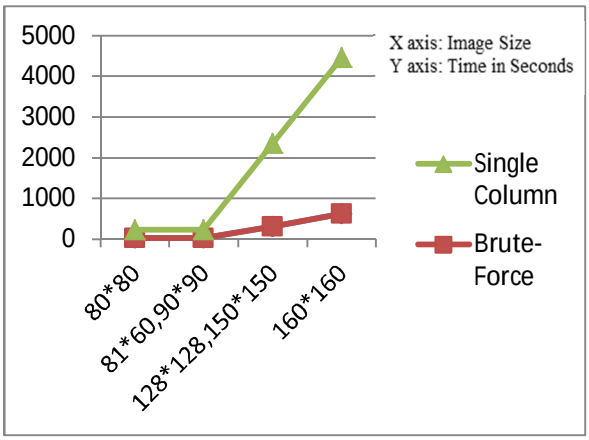

Fig 3: Graph denoting time computation of Single Column and Brute-Force from above table for different image size

From the graph, we can note that computation time drastically increases as the image size increases in

case of brute-force approach compared to single column approach that uses sequential overlap concept.

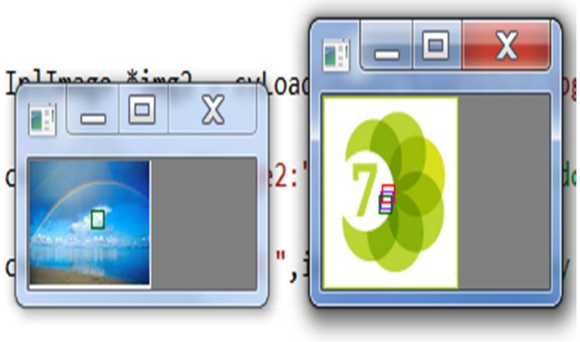

Fig 4: Output for patch size r = 8*8, for source image of size 81*60 and target image of size 90*90

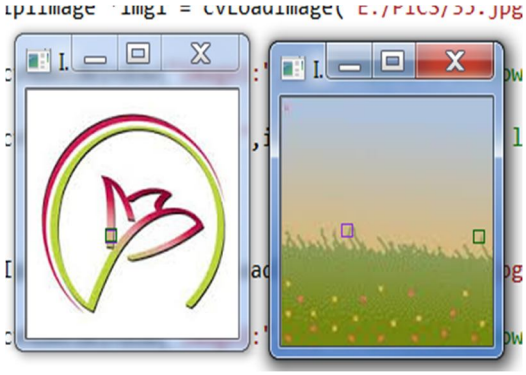

Fig 5: Output for patch size r = 8*8, for source image of size 160*160 and target image of size 160*160

In above figures, represented by small squares are nearest neighbours found between source and target images when compared.

II. Time Computation in Different Image Sizes in Single Column and Multiple Column

Image Size(Source Image, Target Image)	Execution time of Multiple Columns in seconds	Execution time of Single Column in seconds
100*100	72	128
150*150	314	475
200*200	890	1574
200*200,160*160	619	1003

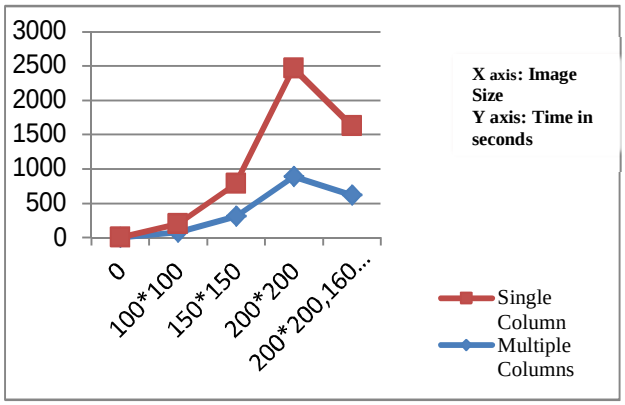

Fig 6: Graph denoting time computation of Multiple columns and Single Column from above table for different image size

From the graph, we can note that computation time is less when multiple processing approach is used compared to single column processing approach.

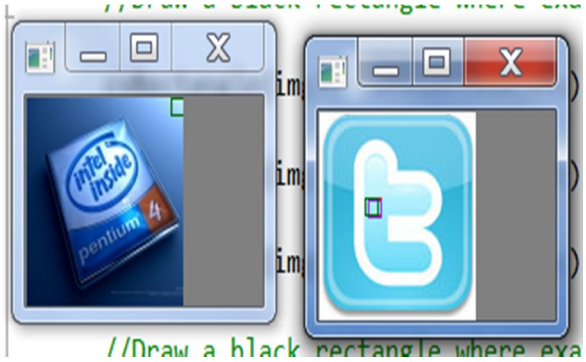

Fig 7: Output for patch size r = 8*8, for source image of size 100*100 and target image of size 100*100

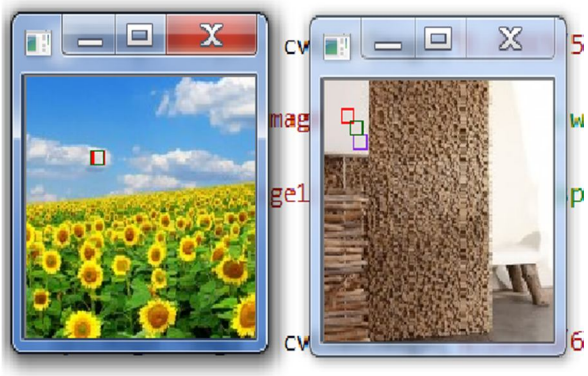

Fig 8: Output for patch size r = 8*8, for source image of size 150*150 and target image of size 150*150

III. Time Computation in Different Patch Sizes in Single Column and Multiple Column

Patch Size(Patch Width, Patch Height)	Execution time of Multiple Columns in seconds	Execution time of Single Column in seconds
4*4	209	299
8*8	309	567
12*12	435	780
16*16	488	876
20*20	549	1004

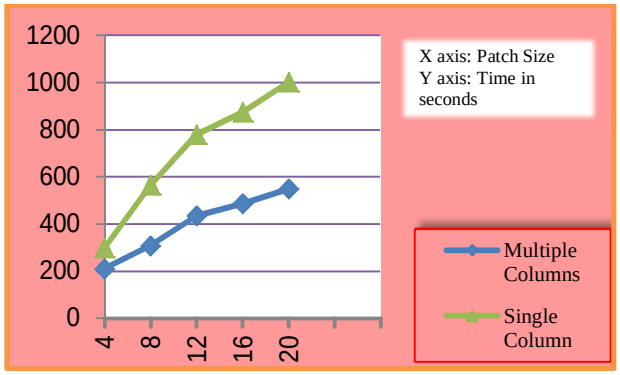

Fig 9: Graph denoting time computation of Multiple Column and Single Column from above table for different patch size

Above graph denotes computation time for different patch sizes using multiple column processing and single column approach. Computation time is reduced when multiple column processing approach is used for different patch size than single column approach.

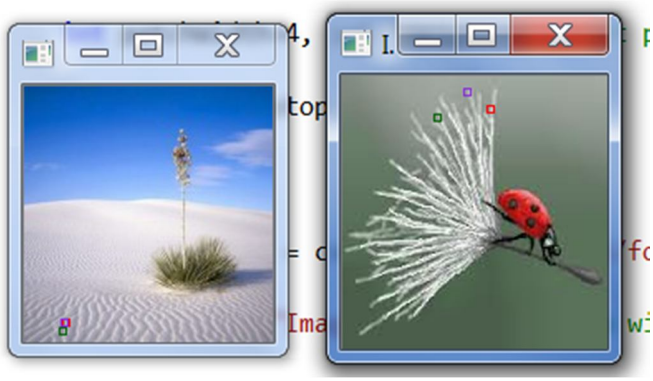

Fig 10: Output for patch size r = 4*4, for source image of size 150*150 and target image of size 160*160

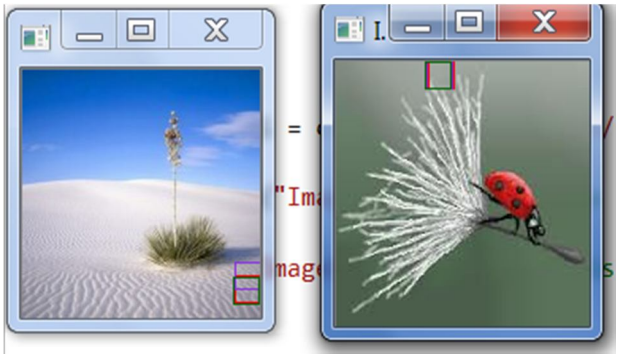

Fig 11: Output for patch size r = 16*16, for source image of size 150*150 and target image of size 160*160

V. Conclusions and Future Enhancement

A novel fast exact patch matching algorithm is proposed. This does not require reconstruction of hierarchical data structure. As adjacent patch pairs overlap to certain extent, redundant calculations are eliminated, reducing the time complexity from $O(mr^2)$ to $O(mr)$. Memory requirement needed is also minimal. Relying on brute-force comparison, match found is guaranteed to be exact. Each patch pair is compared only once. By processing multiple columns at a time, patch matching is further accelerated.

Although proposed method for image is significantly faster, to process extremely large image with a large patch size, the efficiency has to be further improved for the interactive image processing and editing. With the rapid development of the graphics hardware, efficiency could be achieved in the near future.

REFERENCES

[1] Chunxia Xiao, Meng Liu, Yongwei Nie, and Zhao Dong ,"Fast Exact Nearest Patch Matching for Patch-Based Image Editing and Processing", IEEE trans visualization and computer graphics, vol. 17, no. 8, August 2011.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "Patchmatch: A Randomized Correspondence Algorithm for Structural Image Editing",Proc. ACM SIGGRAPH, 2009.

[3] N. Kumar, L. Zhang, and S. Nayar, "What is a Good Nearest Neighbors Algorithm for Finding Similar Patches in Images?"Proc. European Conf. Computer Vision (ECCV), pp. 364-378, 2008.

[3] G. Shakhnarovich, T. Darrell, and P. Indyk, Nearest-Neighbor Methods in Learning and Vision: Theory and Practice. MIT Press, 2005.

[4] B. Weiss, "Fast Median and Bilateral Filtering",ACM Trans.Graphics, vol. 25, pp. 519-526, 2006.

[5] E. Chavez, G. Navarro, R. Baeza-Yates, and J. Marroquin, "Searching in Metric Spaces," ACM Computing Surveys, vol. 33, no. 3, pp. 273-321, 2001.

[6] S. Arya, D.M. Mount, N.S. Netanyahu, and R. Silverman, and A.Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching",J. ACM, vol. 45, no. 6, pp. 891-923, 1998.

[7] T. Huang, Two-Dimensional Signal Processing II: Transforms and Median Filters, Springer-Verlag, 1981.

[8] J. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," Comm. ACM, vol. 18, no. 9, pp. 509-517,1975.