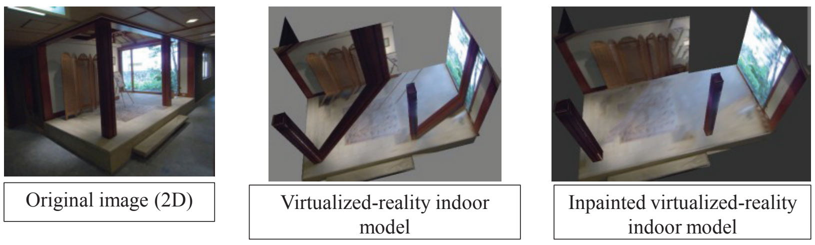
the International Display Workshops (IDW/AD'12), Vol.19, No.3, pp.1479-1482 (2012)

Texture Significant Hash Function with Robust Occlusion Handling for Fast Inpainting the Virtualized-Reality Models

Kalaivani Thangamani*+, Tomoya Ishikawa*, Koji Makita*, Takeshi Kurata*+

+Service engineering and mixed reality laboratory, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, 305-8577, Japan.
*National institute for advanced industrial science and technology, Center for service research, 1-1-1 Umezono, Tsukuba, 305-8568, Japan.

Keywords: Virtualized-Reality models, Inpainting, Hash tables, Exemplar-Based inpainting


Original image (2D)
Virtualized-reality indoor model
Inpainted virtualized-reality indoor model

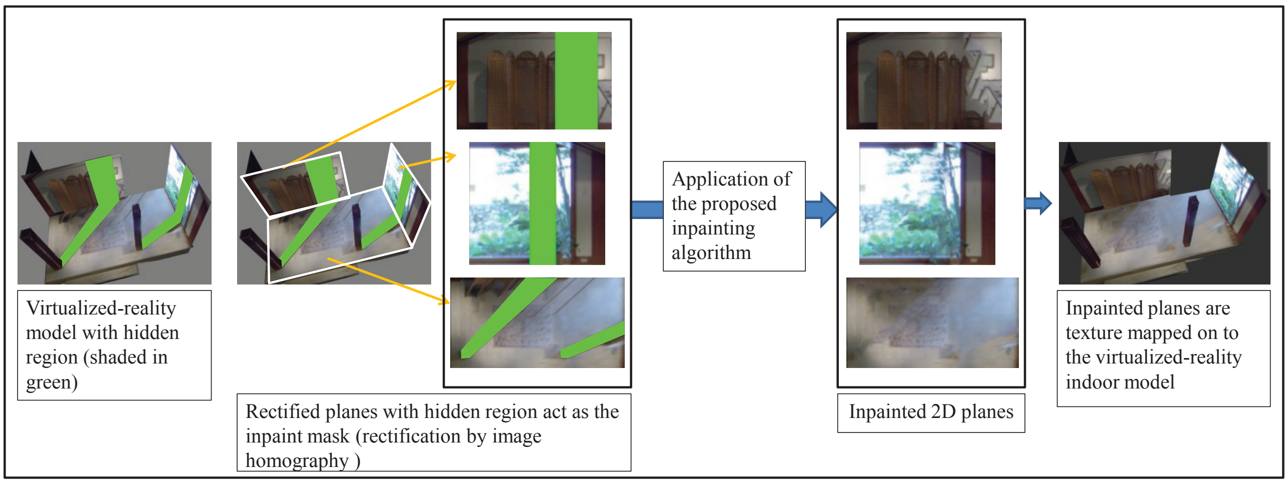

Virtualized-reality model with hidden region (shaded in green)
Rectified planes with hidden region act as the inpaint mask (rectification by image homography)
Application of the proposed inpainting algorithm
Inpainted 2D planes
Inpainted planes are texture mapped on to the virtualized-reality indoor model

Fig. 1 Inpainting for the virtualized-reality indoor modeling

ABSTRACT

This paper discusses the texture significant hash table usage for speed up the inpainting process in the virtualized-reality indoor models. The proposed hash works are included in the Exemplar-Based inpainting and this revised method is tested in the virtualized-reality indoor models. The novelty of this paper lies in the hash function design and the occlusion handling measures. Gray level co-occurrence matrix (GLCM) is used for designing the texture significance hash function. The time consuming stage in the Exemplar-Based inpainting is the repeated texture patch searching process. The proposed hash function reduces the patch selection to minimum/single search.

1. INTRODUCTION

Inpainting, the technique used for modifying /repairing the image parts finds its application in many fields. This paper discusses one such application which is to improve the textures in the virtualized-reality indoor models. Virtualized-reality model helps to turn the real world in to the virtual one, by using the clues such as photos from the real world. The 3D modeler [1,2] allows the user to create the indoor environment of their desired environment from a single or multiple photos by using simple interaction techniques based on computer vision principles. Projective texture mapping is used in the 3D modeler for mapping the textures over the 3D planes. There are often untextured regions on some of the 3D planes since it is not easy to take a set of photos so that every region of the 3D model is included at least in one of those photos. Exemplar-Based inpainting method [3], known for its fine structure propagation details, is adopted for handling the untextured regions. The computation time becomes a constraint in using this method. It is necessary to reduce the computation time for the efficient post processing in the virtualized-reality indoor modeling.

2. INPAINTING FOR THE VIRTUALIZED-REALITY INDOOR MODELS

Figure 1 shows the example for the virtualized- reality indoor modeling created by the interactive indoor modeler. The virtualized-reality indoor model shown in Fig. 1 is created from the single 2D image. The regions that are not captured in the 2D image will not have textures for their 3D world. Obviously these hidden regions will get the textures of their frontal planes, as the result of the projective texture mapping. These hidden regions are traced automatically by the modeler and these regions act as the inpainting mask. The regions shaded in green in the figure 1 shows the inpainting mask. The 3D planes are rectified and then inpainted by the proposed hash based fast inpainting technique. The inpainted planes are fixed back to the virtualized-reality models. Figure 1 shows the complete flow of the post processing in the virtualized-reality indoor modeling.

3. EXEMPLAR-BASED INPAINTING METHOD

Exemplar-Based inpainting [3] fills the mask region with the iterative patch based inpainting. The patch size determines the quality of the inpainted result and it depends on the texture and structure details of the particular image. The iteration is continued till the entire mask region is filled by the texture patches. The suitable texture patch is selected by calculating the sum of the squared differences (SSD) of the pixel values between the texture patches. During every iteration, the occluded patch is compared with all other patches in the image which counts for thousands and thousands of comparisons. This step counts for the increase in the inpainting time. There comes the necessity to revise the tedious patch searching process.

Hash tables [4] are well known for the quick access of data elements in an array. The following section explains the design of hash function for the patch searching stage.

4. HASH FUNCTION DESIGN

The hash function should be designed in such a way that it allots a unique address for every patch. The principle of hash function is to convert the input to an address of the corresponding hash bin where the input can be stored. So the hash function should be capable of handling the texture patches in terms of numerical values.

Textures are effectively represented by the Haralick features [5] such as the energy, entropy and other subsequent parameters. GLCM is the basic gray level matrix behind the Haralick feature calculation. Figure 2 shows the design of hash function with the GLCM. Due to the similarity in the GLCM structure, only the lower triangular portion of the GLCM is considered. Figure 2 shows the GLCM structure for 8 gray levels forming the 8 rows and 8 columns. Every bin is given a virtual number and the total number of filled bins forms the address for the input patch.

For example, sample patch1 is the quantized image patch (reduced to 8 gray levels), fills the GLCM bins numbered as 1,6,13 and 20, in total 4 bins are filled. So the sample patch1 is stored in the hash bin whose address is 4. This step groups the similar gray level patches into the single bin. The effective design would be to store each patch in an unique address which is an advancement from our previous contribution [6].

The whole patch is divided in to 4 sub quarters and they are stored in ascending/descending order of their dominant bin entry. For every group of patches in the particular hash bin, there will be one gray level that holds the highest value and the bin that holds this gray level is called as the dominant bin. The patches are partitioned in order to have the robust occlusion handling. The number of chained hash entries is equal to the number of different GLCM combinations. The sub quarters are stored in the subsequent chains of every hash bin, and the linked list [7] serves the best connection between the chained combinations. This stage helps to handle the occluded patch comparison.

5. HASH TABLE QUERY

Figure 3 explains the query process with the hash table. The hash function maps the hash bin for the occluded patch. The occluded patch is divided into four quarters and the un-occluded quarter is considered for the search over the chained hash entries. According to the GLCM combination of the un-occluded patch, the search is transferred to the corresponding chained entry. Then the address nearer to the dominant bin count of the un-occluded quarter is considered as the needed patch. Once this sub-quarter is found out, the corresponding whole patch is retrieved for the patch filling process in the Exemplar-Based inpainting method.

6. BLENDING THE HASH WORKS WITH THE EXEMPLAR-BASED METHOD

The hash works are referred in the patch searching stage of the Exemplar-Based inpainting method. The hash table is made before the inpainting begins. Once the priority occluded patch is found out by the Exemplar-Based method, the occluded patch queries the hash table for the matching texture patch. This is the step which counts for the drastic reduction of the computation time.

The normal Exemplar-Based inpainting takes N searches for every patch fill, (N represents the eligible texture patches which are free from occlusion) whereas the proposed hash works reduces the patch search to single/minimum search.

7. RESULTS AND DISCUSSION

The proposed hash-based inpainting is tested successfully in the planes of the virtual reality model. The inpainted images are subjected to qualitative, quantitative and subjective analysis.

The quantitative analysis is given in table 1. The indoor model of our office, a Japanese restaurant and the ISMAR2009 site are taken for testing the hash based inpainting. The time taken for inpainting the planes with the Exemplar-Based inpainting is compared against the hash table included Exemplar-Based inpainting. The time shown in the last column of the table 1 is the total time that included the hash table making and the inpainting process. From the table, it is evident that the computation time is drastically reduced with the help of the hash works.

The GLCM is tested with various gray levels such as 8, 16, 32 etc., and the inpainted results show only trivial difference between the gray levels. These properties show the robustness of the designed hash function.

8. CONCLUSION AND FUTURE WORK

The texture significant hash works are discussed for improving the performance of the Exemplar-Based inpainting method.

The future work will extend the hash table to the next dimension which will hold the entire texture patches from the 3D model. The extended hash table would be useful in inpainting the planes that have limited or distorted textures.

The plane orientation is considered for separating and grouping the textures in the extended hash table. The hash table may be designed to hold the texture patches with different orientations to handle the curved structures. These necessities and challenges encourage the future works.

ACKNOWLEDGEMENT

This work was supported by Strategic Japanese-French Cooperative Program on Information and Communications Technology Including Computer Sciences (ANR and JST)

REFERENCES

[1] T. Ishikawa, K. Thangamani, M. Kourogi, A. P. Gee, W. M. Cuevas, K. Jung and T. Kurata, “In-Situ 3D indoor modeler with a camera and self contained sensors”, HCII2009, San Diego, CA, LNCS, Vol. 5622, pp. 454-464, (2009).

[2] T. Kurata, M. Kourogi, T. Ishikawa, J. Hyun and A. Park, “Service cooperation and co-creative intelligence cycles based on mixed-reality technology”, Proceedings of INDIN 2010, pp. 967-972 (2010).

[3] A. Criminisi, P. Perez and K. Toyama, “Object removal by Exemplar-Based inpainting”, IEEE CVPR, Vol. 2, pp. 721-728, (2003).

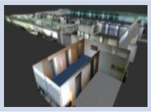
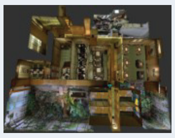
[4] Thomas H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, “Introduction to algorithms”, MIT press, second edition.

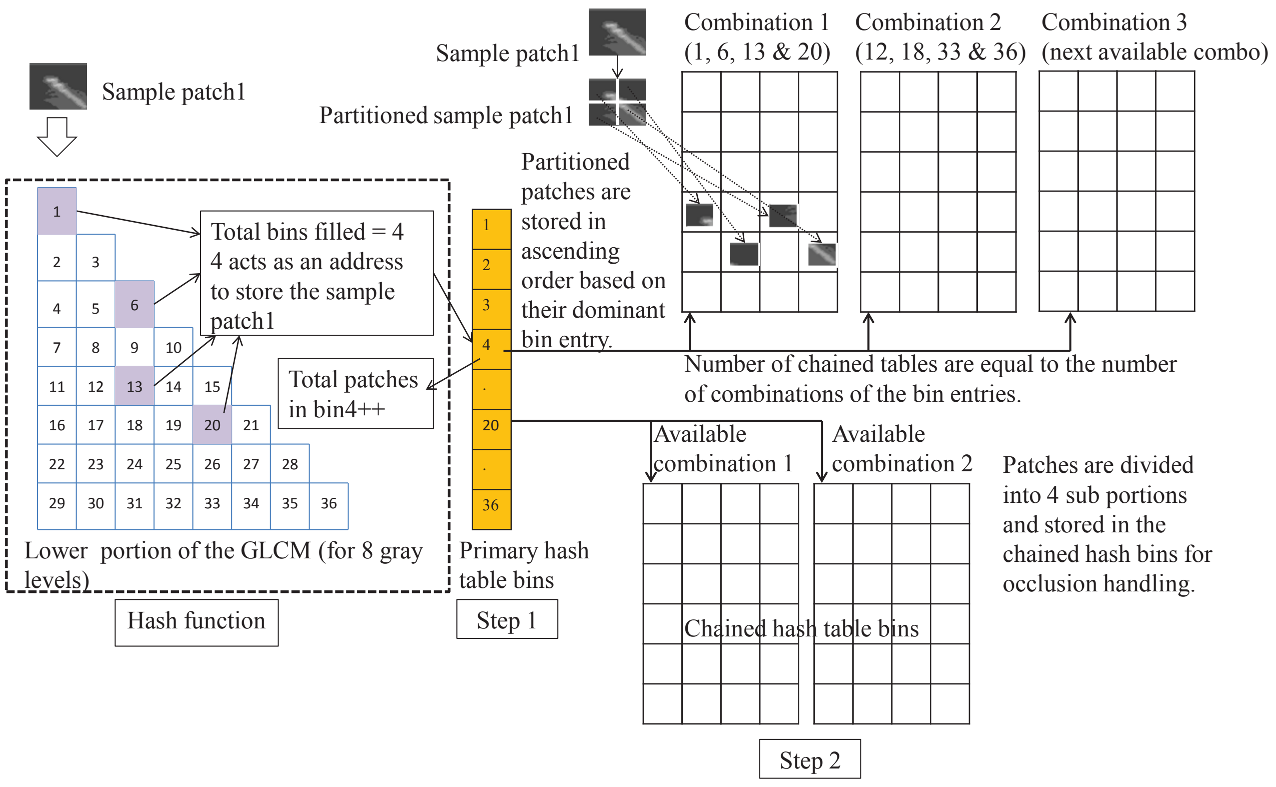
[5] R.M. Haralick, K. Shanmugam and I. Dinstein, “Textural features for image classification”, IEEE Transactions on systems, man, cybernetics, Vol. SMC-3, No. 6, pp. 610-621, (1973).

[6] K. Thangamani, T. Ishikawa, K. Makita and T. Kurata, “Fast and texture-structure preservable inpainting and its application for creating virtualized-reality indoor models”, International journal of computer science and informatics, ISSN: 2231-5292, Vol. 1, issue 3, pp. 85-101, (2012).

[7] E. Miyamoto and T. Merryman Jr, “Fast calculation of Haralick texture features”, Human computer interaction institute, Carnegie Mellon University, Pittsburgh, USA.

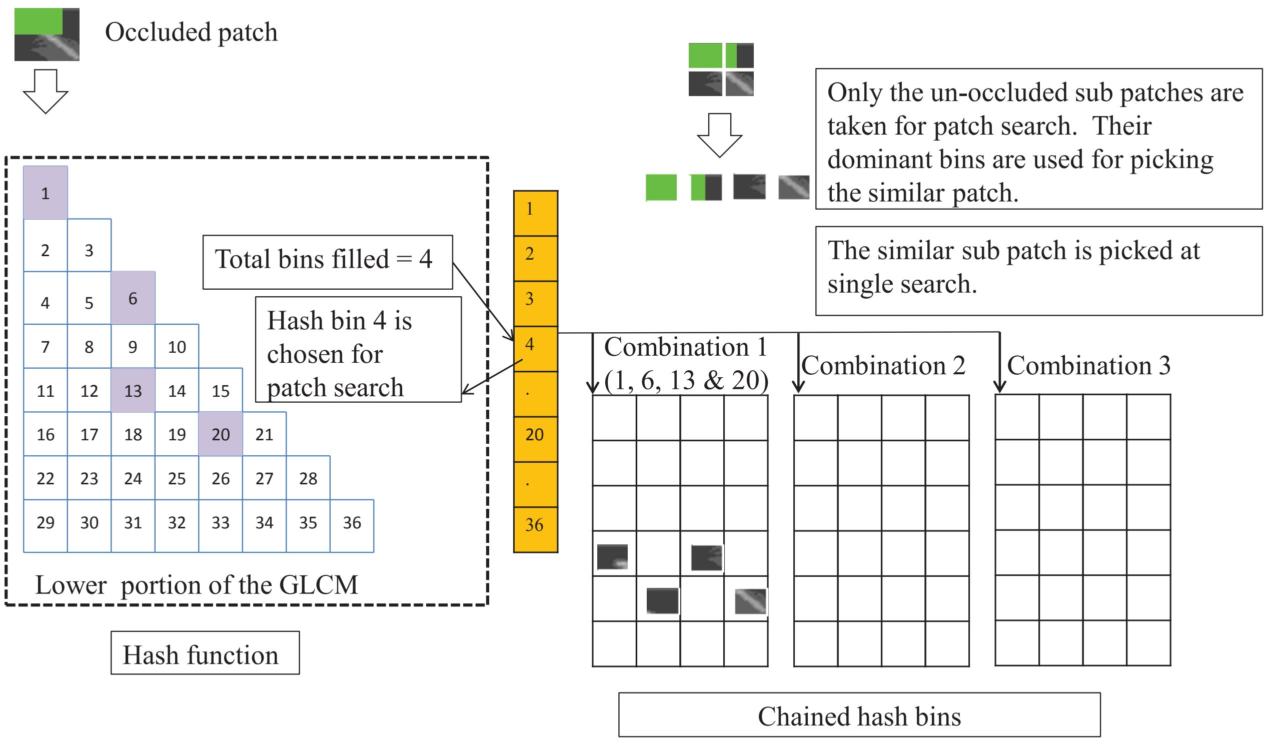
Table 1. Quantitative analysis

Virtualized-Reality indoor model	# of photos used	# of 3D planes used	# of 3D planes that need inpainting	Time taken for Exemplar-Based inpainting	Time taken for hash based inpainting (time includes the making of hash table)
office 	68	331	102	3.2 hrs	12.5 min
Japanese restaurant 	68	425	89	2.8 hrs	7.2 min
ISMAR2009 site	57	333	62	1.2 hrs	5.3 min



Sample patch1
Sample patch1
Partitioned sample patch1
Combination 1 (1, 6, 13 & 20)
Combination 2 (12, 18, 33 & 36)
Combination 3 (next available combo)
Partitioned patches are stored in ascending order based on their dominant bin entry.
Total bins filled = 4
4 acts as an address to store the sample patch1
Total patches in bin4++
Number of chained tables are equal to the number of combinations of the bin entries.
Available combination 1
Available combination 2
Patches are divided into 4 sub portions and stored in the chained hash bins for occlusion handling.
Lower portion of the GLCM (for 8 gray levels)
Primary hash table bins
Chained hash table bins
Hash function
Step 1
Step 2

Fig. 2 Hash table storage



Occluded patch
Only the un-occluded sub patches are taken for patch search. Their dominant bins are used for picking the similar patch.
The similar sub patch is picked at single search.
Total bins filled = 4
Hash bin 4 is chosen for patch search
Combination 1 (1, 6, 13 & 20)
Combination 2
Combination 3
Lower portion of the GLCM
Hash function
Chained hash bins

Fig. 3 Hash table query