

SPATIO-TEMPORAL TEXTURE SYNTHESIS AND IMAGE INPAINTING FOR VIDEO APPLICATIONS

Sanjeev Kumar, Mainak Biswas, Serge J. Belongie and Truong Q. Nguyen

University of California, San Diego

ABSTRACT

In this paper we investigate the application of texture synthesis and image inpainting techniques for video applications. Working in the non-parametric framework, we use 3D patches for matching and copying. This ensures temporal continuity to some extent which is not possible to obtain by working with individual frames. Since, in present application, patches might contain arbitrary shaped and multiple disconnected holes, fast fourier transform (FFT) and summed area table based sum of squared difference (SSD) calculation [1] cannot be used. We propose a modification of above scheme which allows its use in present application. This results in significant gain of efficiency since search space is typically huge for video applications.

1. INTRODUCTION

Any semblance of order in an observation is a manifestation of redundancy of its present representation. Such redundancies have been exploited for different purposes in innumerable applications, one canonical example being data compression. Specifically, in video compression, motion compensation and transform coding are used to exploit the spatio-temporal redundancy in the video signal. But the types of redundancies exploited by current methods are rather limited. This is exemplified by the success of error concealment techniques. Moreover, Discrete cosine transform (DCT), the most commonly used transform coding technique in video compression, has been found to be effective only for small block sizes, which shows its inability to exploit redundancies extending upto larger extent. Wavelet has been relatively more successful in this respect for certain classes of images but hasn't found much use in general purpose video compression.

It is interesting to compare aforementioned techniques with texture synthesis and image inpainting both of which also assume the existence of redundancies in images and video, but exploit it for other purposes different from compression. The type of redundancy exploited by these techniques e.g.

texture synthesis is somewhat complementary to DCT etc. Texture synthesis works on more global level. The motivation behind present work was the perceptual quality of output of texture synthesis and image inpainting methods using little amount of original data. Intra block prediction and context based arithmetic coding in H.264, try to exploit some redundancies which are not exploited by DCT alone, but these still work on a very local level and fail to exploit redundancies at more global level. Although, building a practical video codec based on these principles would require more research effort, present work is intended to be a small step in that direction.

Our approach is similar to [2] which uses non-parametric texture synthesis approach for image inpainting with appropriate choice of fill order. The contribution of present work is two fold. First, we extend the work of [2] to three dimensions, which is more natural setting for video. Second, we extend the work of [1], which allows to use FFT based SSD calculation for all possible translations of a rectangular patch, to arbitrary shaped regions, thus making it applicable to image inpainting.

This paper is organised as follows. In section 2 we briefly review the relevant literature. Section 3 describes the proposed approach. Experimental results have been presented in Section 4. Some future research directions are discussed in Section 5.

2. PREVIOUS WORK

Related prior work can be broadly classified into following three categories.

1. **Texture Synthesis:** Texture synthesis has been used in the literature to fill large image regions with texture pattern similar to given sample. Methods used for this purpose range from parametric, which estimate parametrized model for the texture and use it for synthesis, e.g. Heeger et al. [3], to nonparametric, in which synthesis is based on direct sampling of the supplied texture pattern, e.g. Efros and Leung [4]. Texture synthesis methods have also been used to fill in small holes in the image which might originate due to deterioration of image or desire to remove some

This work is supported in part by NSC and by CWC with matching grant from UC DiMi.

objects. But, aforementioned texture synthesis methods have been found to work poorly for highly structured textures which are common in natural images. Graphcut based techniques [5] try to maintain structural continuity during texture synthesis by finding optimal seam while copying texture patches from different portions of image.

2. **Image Inpainting:** Image inpainting has been used in the literature to fill in small holes in the image, by propagating structure information from image to the region to be filled. Typically, diffusion is used to propagate linear structure based on partial differential equation [6] [7]. These are found to perform well in filling small holes but produce noticeable blur when filling large holes. Recently, Criminisi et al. [2] proposed a method which combines the benefit provided by texture synthesis and image inpainting. The results of their algorithm compare favorably with other state of the art in the field without resorting to texture segmentation or explicit structure propagation and is able to fill in large holes. Our approach is similar to their work but we investigate some issues which are not so important for static images but become relevant for videos.
3. **Applications in Video:** Some contemporary applications of these techniques include recovery of lost data in image sequences [8], video logo removal [9], object removal [2].

3. PROPOSED APPROACH

In this section, we begin with brief recapitulation of the notation and algorithm presented in [2]. For more details, reader should refer to the original manuscript.

3.1. Problem Statement and Some Notations

Given an image or video $\mathcal{I} = \Phi \cup \Omega$, where Φ is the source region, set of pixels whose value is known and Ω is the target region (hole), set of pixels whose value is to be filled. $\delta\Omega$ represents boundary between source and target region, which is a contour in 2D and a surface in 3D. For all boundary pixels $\mathbf{p} \in \delta\Omega$, ψ_p denotes a patch centered at \mathbf{p} .

3.2. Inpainting in 2D

We assign a confidence term to every pixel on the boundary, which is given by

$$C(p) = \frac{\sum_{q \in \psi_p \cap (\mathcal{I} - \Omega)} C(q)}{|\psi_p|} \quad (1)$$

Initially, $C(p) = 0, \forall p \in \Omega$. and $C(p) = 1, \forall p \in \mathcal{I} - \Omega$. We also assign a data term to every pixel on the boundary, which is given by

$$D(p) = |\nabla I_p^\perp \cdot n_p| \quad (2)$$

where, ∇I_p represents gradient of the image and n_p represents normal vector to the region boundary. Based on data and confidence terms a priority is assigned to every pixel on the boundary, which is given by $P(p) = C(p) * D(p)$. The patch centered at pixel with maximum priority value $\hat{p} = \arg \max_p P_p$ is selected as the starting point for filling. Search is performed over all patches $\psi_q \in \Phi$ for best matching patch $\psi_{\hat{q}}$ according to modified sum of squared difference criterion $d(\psi_{\hat{p}}, \psi_q)$ which includes only those pixels of $\psi_{\hat{p}}$ which are already filled in. Authors of [2] performed search in CIE Lab color space, but in present work we only deal with grayscale images. But all the algorithms presented here can easily be extended to any color space. Values for all pixels $r \in \psi_{\hat{p}} \cap \Omega$ are copied from corresponding pixels in $\psi_{\hat{q}}$ and confidence terms are copied from \hat{p} . Data term is recomputed for pixels on the boundary created by recent fill and above steps are repeated until all the pixels get filled. The algorithm described so far is similar to that presented in [2] except lack of a normalization factor in (2), which doesn't affect the final result. Now we describe contribution of the present work in following two subsections.

3.3. Extension to 3D

A naive approach to extend the algorithm described above to video would be to treat video as a collection of frames or images and perform the same operation on each frame independently. There are two main disadvantages of this approach. Because of temporal correlation among the frames a matching patch is also likely to be found in temporally adjacent frames especially if a portion of the region to be filled (and hence absent) in one frame is present in some other neighboring frame. The other disadvantage is that even if every filled frame is spatially coherent, temporal coherence is not ensured, which might result in visible artifacts in the video. For video texture synthesis, 3D spatio temporal patches were used in [5]. Similarly, we propose to use 3D spatio temporal patches in the present framework. This ensures temporal consistency in the video frame because of the similar reasons which result in spatial consistency to be maintained in 2D version of the present algorithm.

In order to extend the present algorithm to 3D, we need generalizations of confidence term of (1) and data term of (2). Generalization of confidence term is trivial since now we just need to sum over pixels in the 3D patch. Generalization of data term is not that obvious. Although we could still compute the spatio-temporal gradient of image intensity (∇I_p) and normal vector (n_p) to the boundary surface,

there is no unique perpendicular direction to the gradient of image intensity i.e. ∇I_p^\perp is not unique. We propose following modified data term using cross product of vectors.

$$D(p) = |\nabla I_p \times n_p| \quad (3)$$

which is defined for both 2D and 3D, maintains the intuition of propagating structural information and has a nice property of reducing to (2) in 2D. For calculating normal vector (n_p) to boundary surface $\delta\Omega$ we use a different scheme from [2] which is equally valid in 2D and 3D. We maintain a binary mask for representing Ω whose smoothed gradient gives us normal vector (n_p) to boundary surface $\delta\Omega$.

3.4. FFT based search for arbitrary shaped patch

Searching for best matching patch is the most computationally demanding part of the present algorithm. Possibility of real time implementation critically depends on the efficiency of this search. In this subsection, we revisit the problem of searching for matching patch, describe an algorithm presented in [1] which solves a closely related problem efficiently but is not directly applicable in the present situation, propose a modification of above algorithm which makes it applicable in the present situation.

We look for the best match for the selected patch $\Psi_{\hat{p}}$ in the search range Φ . Equivalently, we look for translations of the patch such that the portion of the patch overlapping the surrounding region matches it well - only those translations that allow complete overlap of the patch with the surrounding region are considered. The cost of a translation of the patch is now defined as:

$$C(t) = \sum_{p \in \Psi_{\hat{p}} \cap (\mathcal{I} - \Omega)} |\ell(p-t) - \ell(p)|^2 \quad (4)$$

where, $\ell(p)$ denotes intensity or color value at pixel \mathbf{p} . The SSD-based search described in (4) can be computationally expensive if the search is carried out naively. Computing the cost $C(t)$ for all valid translations is $O(n^2)$ where n is the number of pixels in the image or video. If the domain of summation is simple rectangular region, the search can be accelerated using Fast Fourier Transforms(FFT) [1]. We can rewrite (4) as:

$$C(t) = \sum_{p \in \Psi_{\hat{p}} \cap (\mathcal{I} - \Omega)} \ell(p-t)^2 - 2\ell(p-t)\ell(p) + \ell(p)^2 \quad (5)$$

Third term is independent of t and can be discarded from consideration while minimizing over t . The first term in (5) is sum of squares of pixel values over search region around the patch. For non masked region it can be computed efficiently in $O(n)$ time using summed-area tables [10](also known as integral image). The second term is a convolution of the patch with the image search region and for non-masked region can be computed in $O(n \log(n))$ time using

FFT. But, summation domain in (5) is not a simple rectangular one and can contain arbitrary shaped and disconnected holes and hence algorithm presented in [1] is not applicable here. We propose following modification of aforementioned algorithm to make it applicable in the present situation.

First term consists of summation of square of intensities of pixels of image which correspond to filled pixels of patch. For different values of translation t this summation can be expressed as convolution of “squared” image with appropriate binary mask and hence can be efficiently calculated using FFT as shown below.

$$\begin{aligned} \sum_{p \in \Psi_{\hat{p}} \cap (\mathcal{I} - \Omega)} \ell(p-t)^2 &= \sum_{p \in \Psi_{\hat{p}}^r} \ell(p-t)^2 m(p) \quad (6) \\ &\sim \ell(p)^2 \otimes m(p) \quad (7) \end{aligned}$$

where, $\Psi_{\hat{p}}^r$ denotes rectangular (cuboid for 3D patch) bounding box of $\Psi_{\hat{p}}$. $m(p)$ denotes binary mask corresponding to region $\Psi_{\hat{p}} \cap (\mathcal{I} - \Omega)$, and \otimes denotes convolution. Although complexity of this step is $O(n \log(n))$ as opposed to $O(n)$ complexity of summed area table, this doesn't affect the overall complexity of search as explained below.

For evaluating second term, the presence of the filled and the unfilled pixels in the patch prohibits straight forward implementation of the convolution. This problem can be circumvented by assuming the masked region of the patch does not contribute to the convolution sum and unknown pixel values in the image are not allowed to appear in convolution sum. In practice the unfilled pixels in patch are set to zero, unfilled pixels in image are set to infinity and the convolution can be evaluated as product of the Fourier transforms of the two terms.

$$\begin{aligned} \sum_{p \in \Psi_{\hat{p}} \cap (\mathcal{I} - \Omega)} \ell(p-t)\ell(p) &= \sum_{p \in \Psi_{\hat{p}}^r} \ell_\infty(p-t)\ell_0(p) \quad (8) \\ &\sim \ell_\infty(p) \otimes \ell_0(p) \quad (9) \end{aligned}$$

where, $\ell_\infty(p)$ and $\ell_0(p)$ denote modified image and patch by setting unknown pixels to ∞ and 0 respectively. Complexity of computing this term for arbitrary shaped patches remains the same as that for rectangular patches i.e. $O(n \log(n))$. Thus, overall asymptotic complexity of arbitrary shaped patch matching is same as that for rectangular patches.

This technique is also applicable for applications which perform block based motion estimation and there is a binary mask associated e.g. for delimiting the region of interest or object boundaries.

4. RESULTS

Fig. 1 shows one frame of the video sequence used for experiments. Fig. 2 shows the result of inpainting algorithm for various cases. Additional results are available at <http://videoprocessing.ucsd.edu/~sakumar/inpainting.html>.



Fig. 1. One frame of original video sequence used for experiment



Fig. 2. Input images to and output images from the inpainting algorithm have been shown on left and right respectively (only relevant portion shown). First row: Hole is 2D and search space is 2D. Algorithm works well because of significant edge information to be propagated. Second row: Hole is 2D and search space is 2D. Algorithm doesn't work well because of absence of significant edge information to be propagated. Third row: Hole is 2D and search space is 3D. Algorithm works well because of availability of enough information in the temporal neighbors to fill the missing part.

5. DISCUSSION AND FUTURE WORK

Unlike image restoration and other applications of image inpainting, automatic patch size selection is crucial for video applications. Dynamic nature of scene content precludes use of a fixed patch size. For this purpose, automatic scale selection methods from scale-space theory might be useful. For a practical video codec, inpainting alone may not be sufficient as it assumes no data is available in the region to be filled, thus making it difficult to have any control over rate-distortion performance. Some hybrid algorithm of inpainting and restoration should be more useful.

Non-parametric nature of present work makes it susceptible to lack of sufficient amount of data. Even though a single frame of high resolution video might contain millions of pixel, for 243 dimensional space of $9 \times 9 \times 3$ patch million pixels are very small sample, and asymptotic performance guarantees of non-parametric method of probability density estimation doesn't apply. This can be alleviated to some extent by using semiparametric approaches such as video epitomes (<http://www.psi.toronto.edu/computerVision.html>).

6. REFERENCES

- [1] S. L. Kiltathau, M. Drew, and T. Moller, "Full search content independent block matching based on fast fourier transform," in *Proc. ICIP (IEEE International Conference on Image Processing)*, 2002, pp. I-669-672.
- [2] 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, Sept. 2004.
- [3] David J. Heeger and James R. Bergen, "Pyramid-based texture analysis/synthesis," in *Proc. ACM-SIGGRAPH*, 1995, pp. 229-238.
- [4] A. Efros and T.K. Leung, "Texture synthesis by non-parametric sampling," in *Proc. ICCV (IEEE International Conference on Computer Vision)*, 1999, pp. 1033-1038.
- [5] V. Kwatra et al., "Graphcut textures: Image and video synthesis using graph cuts," in *Proc. ACM-SIGGRAPH*, 2003.
- [6] M. Bertalmio et al., "Image inpainting," in *Proc. ACM-SIGGRAPH*, 2000, pp. 417-424.
- [7] T. F. Chan and J. Shen, "Variational image inpainting," <ftp://ftp.math.ucla.edu/pub/camreport/cam02-63.pdf>.
- [8] S.D. Rane, G. Sapiro, and M. Bertalmio, "Structure and texture filling-in of missing image blocks in wireless transmission and compression applications," *IEEE Transactions on Image Processing*, vol. 12, no. 3, pp. 296-303, 2003.
- [9] Wei-Qi Yan and M.S. Kankanhalli, "Erasing video logos based on image inpainting," in *Proc. ICME (IEEE International Conference on Multimedia and Expo)*, 2002, pp. 521-524.
- [10] F. C. Crow, "Summed-area tables for texture mapping," in *Proc. 11th Annual Conference on Computer Graphics and Interactive Techniques*, 1984, pp. 207-212.