British Journal of Mathematics & Computer Science
4(20): 2872-2883, 2014

SCIENCEDOMAIN *international*
www.sciencedomain.org

Dynamic Video Texture Inpainting Using Improving LDS

Mohsen Soryani[1*], Amanna Ghanbari[1] and Abbas Koochari[1]

[1]*School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.*

Original Research Article

Received: 04 May 2013
Accepted: 20 February 2014
Published: 27 July 2014

Abstract

Inpainting or completion is used with the aim of restoring damaged regions in images and video frames using the safe regions. This paper introduces a novel approach based on LDS (Linear Dynamic Systems) for inpainting of corrupted video frames which include dynamic textures. In this work, a mask is defined for each frame corresponding to the damaged portion of the frame; the mask determines the target region that will be completed by the proposed method. Notice that the mask is moving. To verify whether corrupted frames are reconstructed correctly or not, a measure named MS-SSIM (Multi-Scale Structural Similarity) is used. The value close to one for this scale introduces more similarity between the two components that are going to be compared. The obtained value for the above mentioned measure is very close to one for our results and the generated video is pleasant.

Keywords: Video inpainting, dynamic texture, LDS, MS-SSIM.

1 Introduction

One of the interesting studies in image and video processing is inpainting or completion of images and videos which means reconstruction of damaged regions of the image or video using correct regions. It is important in Inpainting not to generate pixels that make the result unpleasant to the viewer.

Several researches have been done on video inpainting. In the first effort for video inpainting, an image inpainting method was distinctly applied to all frames of video sequences [1]. Without considering temporal consistency results are not pleasant [2].The first attempt which considered spatio-temporal consistency was stated by [3]. This method uses a cubic patch for completion. Most video inpainting approaches are extensions of image inpainting methods. Patch-based algorithms can be mentioned as one of these extensions. This approach considers the patch with the highest priority in the hole (target region) at each iteration. Then, video data outside the hole is searched with the aim of finding the most appropriate patch that will be used for filling in the hole.

**Corresponding author: soryani@iust.ac.ir;*

Another example is Object-based algorithms that consider larger patches named objects. Clearly, these algorithms inpaint one frame in every iteration. A sample research in this topic is presented in [4] that is an extension of [2].

In [5] a motion layer-based algorithm which distinguishes between motion layers of a video has been proposed. In this method, a video sequence deforms into different motion layers. Therefore, layer order can be determined in overlapping regions. Removing a layer causes to remove the corresponding object. This process makes other layers to contain holes. Filling in these holes results in completing the remaining layers. By superimposing these different layers we will end up with the inpainted video.

Although most researches have addressed static textures, authors in [6] studied dynamic textured background inpainting. This topic has recently become more attractive. In [7] LDS was introduced to model dynamic texture analysis and synthesis. To improve the synthesized video visually, [8] extended the previous work by applying feedback control into LDS, introducing closed-loop LDS.
This paper presents a framework to fill in holes which are extracted from corrupted frames of a video with dynamic textured background. The camera is assumed to be fixed without any motion. The paper has been organized as follows; the proposed method is explained in section 2. Section 3 presents experimental results and then section 4 concludes the paper with a discussion to future works.

2. Experimental Details

Fig. 1 shows a schematic overview of the proposed algorithm

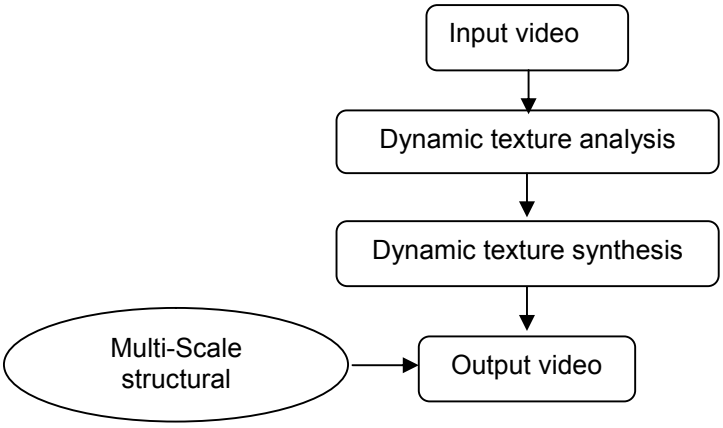


Fig. 1. Schematic overview of the proposed algorithm

To begin with, in order to differentiate between the location of the desired area to be inpainted (target region) and the rest of the frame, a mask is defined. Here, a rectangular mask is considered artificially. Since this green color with RGB values [0 1 0] never happened in the tested video frames, it was used to define the mask. Fig. 2 shows some samples of video frames and their masks.

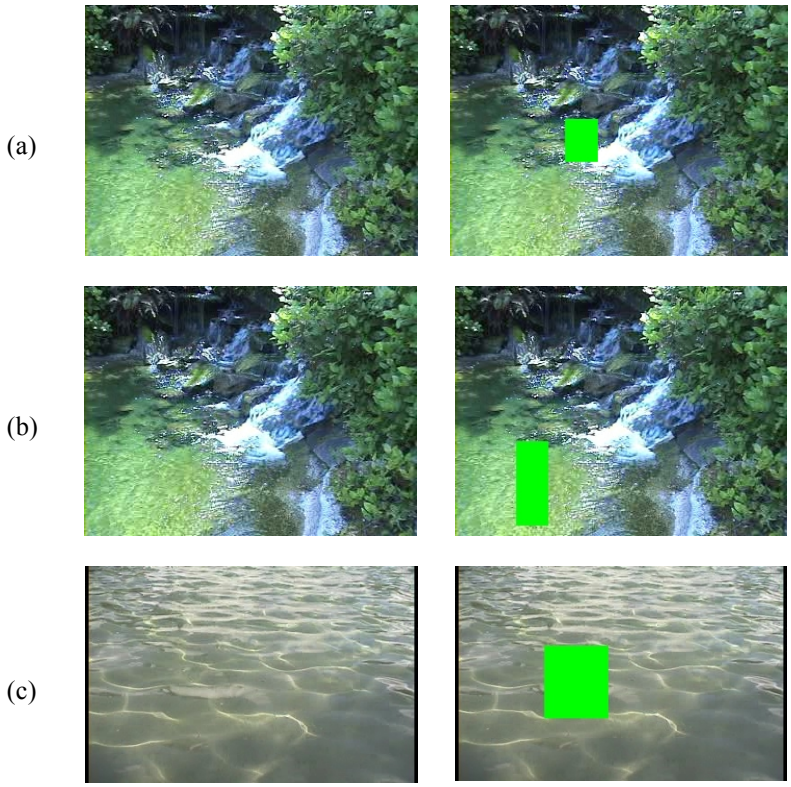


Fig. 2. Video frames and their corresponding masks. One frame has been shown as a sample of each video. (a-b) two masks corresponding to two regions of video have been defined (c) this video frame holds more structure compared to (a-b)

To inpaint dynamic texture, an improved version of texture inpainting by LDS which originally introduced in [9] is used. By superimposing the inpainted part of the damaged frame and the rest of that frame, the completed frame is obtained. The output videos are produced from these completed frames.

2.1 Dynamic Texture Inpainting

In this paper, inpainting is done using LDS. As a result, LDS that was introduced in [9] is firstly restated and then we explain the improvement that is done in this paper.

2.1.1 Linear Dynamic Systems

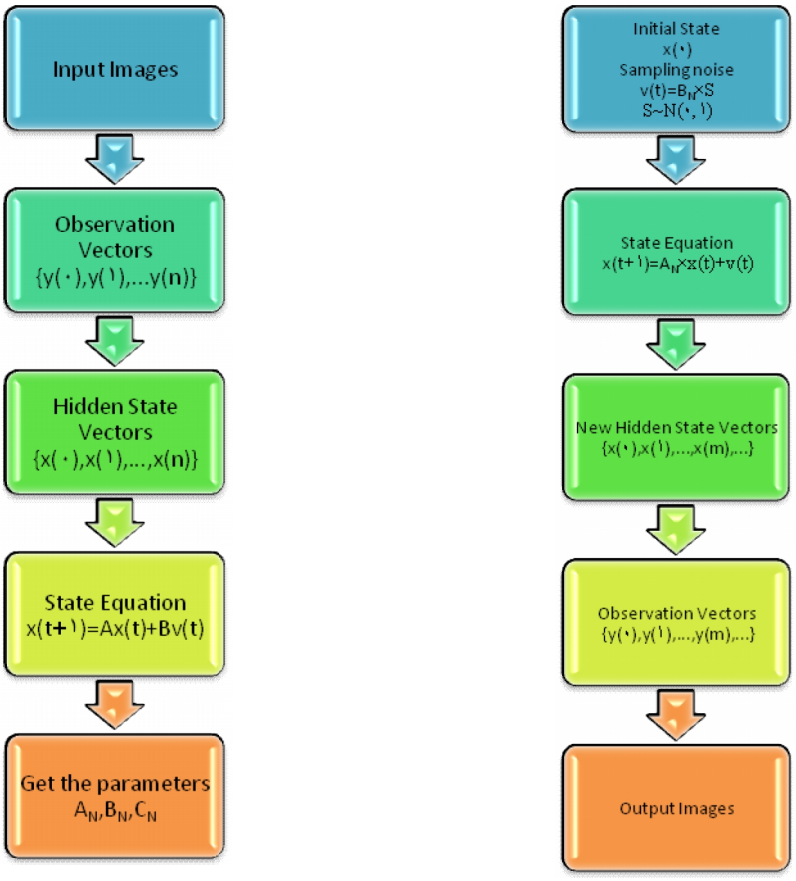
Fig. 3 shows the framework of dynamic texture analysis and synthesis using LDS.

To synthesize dynamic texture, the state-space is represented using the LDS model as follows:

$$x(t+1) = Ax(t) + Bv(t) \tag{1}$$

$$y(t) = Cx(t) \tag{2}$$

Where $x(t)$ is the hidden state vector, $y(t)$ is the observation vector, $v(t)$ is the noise and A, B and C are the system parameter matrices.



(a) (b)

Fig. 3. The framework of dynamic texture analysis and synthesis using LDS [9]. (a) Analysis section (b) Synthesis section

As illustrated in Fig. 3, a finite sequence of images is the input of the linear system and represents the observation vectors $y(t)$ of this system. Using SVD (Singular Value Decomposition) analysis, these images are mapped into $x(t)$ (hidden state vectors) in a lower dimensional space. By this step, a model is learnt for linear dynamic texture synthesis. After that, this model can be used to synthesize new sequences. Using hidden state x_0 as the initial state, the sampling noise $v(t)=B(t)\times S$ ($S\sim N(0,1)$), A_N, the estimated system parameter matrix is utilized with the aim of generation of new state vectors. As mentioned earlier, these attempts are done in lower dimension. As a result, the acquired state vectors are lastly mapped to the high dimensional observation vectors; this step gives a new dynamic texture sequence.

2.1.2 Improving the dynamic texture inpainting

In this paper, temporal information is exploited to synthesize new frames. To be specific, considering frames 10 to 49 as frames including holes (masks) that are going to be completed frames 1 to 10 (10 frames) are synthesized. In other words, frames 1 to 9 are used to generate new frame 10. As it can be seen, paper [9] reproduces frames that we had before. Yet, in this research, we use the model introduced in that paper and produce new frames and add them to what we already have. Having new frame 10, we can replace target region with the reproduced region. Doing this, frame number 10 is completed. The next iteration supplies frame 11 from frames 2 to 10. Frame 10 that is used here is the generated frame, not the original frame. It is continued until the last damaged frame is reconstructed. The following algorithm shows the above description:

- f1 f2 f3 ... fh ... fN are existing frames (fh to fN are frames containing holes)
- for i=h to N
- use fi-h+1 to fi-1 to synthesize f'i-h+1 to f'i
- replace the hole in fi with the region with the same coordinates in f'i
- remove sharp boundaries of inpainted region using a contour-based filter
- calculate MS-SSIM to verify whether the structure of the inpainted frame is preserved or not.

In this research, observation vectors are considered in two situations: 1) the whole frame is considered as one observation vector. 2) The observation vector changes in such a way that only the mask portion of the frame is considered. To be clear, for situation 2, imagine the coordinates of the mask in frame 10 is (110:150,100:130). To inpaint this target region, the same coordinates in frames 1 to 9 are synthesized to generate a new frame. Remember that the mask coordinates change in frame 11. Consequently, the considered region of frames 2 to 10 that are used to synthesize frame 11 will change too. This makes a more accurate synthesis because the rest of the frame does not influence the model of the target region. Since the hole is specified by a special color, the second approach is appropriate and applicable for any video. Moreover, it should be mentioned that large sized videos get more benefit compared to small sized videos that is time saving is another benefit of this method.

Superimposing the reproduced region on the original frame results in a problem; the boundary between the reproduced region and rest of the frame is visible. To reduce this impact, a contour-based algorithm originally used in [10] is applied here. In simpler terms, the contour of the target region is determined. It is not difficult to specify this boundary, because the mask is shown by a selective color. Smoothing is performed just around the boundary using a Gaussian filter.

By now, we can only justify the resulting frames using their appearance. But to have more assurance about the quality of the generated frames, the MS-SSIM measure is used which is explained in the next section.

2.2 Multi-Scale Structural Similarity

MS-SSIM is an objective measure which determines the amount of structure preserved in the inpainted video frames. It should be noted that MS-SSIM is used when the original frames exist. Using this measure, we can examine how much the structure of the original frame and the

inpainted frame are similar to each other. The algorithm introduced originally in [11] and then improved in [12] is used to measure the structural similarity.

2.2.1 SSIM and MS-SSIM

SSIM uses three components; mean, variance and cross-correlation between two patches x and y of two images (i.e. the original image and the reconstructed image) to obtain their structural similarity. If we denote these three components with m(x,y), v(x,y) and r(x,y), respectively, then SSIM will be calculated as follows:

$$\begin{aligned} SSIM(x,y) &= m(x,y)^{\alpha} \times v(x,y)^{\beta} \times r(x,y)^{\gamma} \\ &= \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \right)^{\alpha} \times \left(\frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \right)^{\alpha} \times \left(\frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \right)^{\gamma} \\ &= m \times v \times r \end{aligned} \tag{3}$$

Where μ_x is the mean of x, σ_x is the standard deviation of x, σ_{xy} is the cross-correlation of the mean shifted images x- μ_x and y- μ_y and C_i for i=1,2,3 are small positive constants that are used to prevent division by zero when any divisor is close to zero. α, β and γ are positive values which provide adjustments to the corresponding component's contribution to the overall SSIM value. The original definition for SSIM sets $C_3 = \frac{C_2}{2}$ and $\alpha = \beta = \gamma = 1$. These quantities shorten equation (3) to equation (4).

$$SSIM(x,y) = \left(\frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \right) \times \left(\frac{2\sigma_{xy} + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \right) = m \times (v \times r) \tag{4}$$

There are two points that should be noticed: 1) each two images are divided to some small patches. Then the two patches x and y are selected from the same coordinates of the images X and Y, respectively. 2) The overall SSIM value is calculated as the average of SSIM values computed for each of patch sets x and y.

MS-SSIM subsamples the image K-1 times with the usage of a low-pass filtering which ends to K image scales. Here the variance and cross-correlation are computed for all of the K image scales. Yet, the mean value is only acquired from the coarsest scale K. MS-SSIM is measured using equation (5).

$$MS - SSIM = m_K(X,Y)^{\alpha_K} \prod_{k=1}^{K} v_k(X,Y)^{\beta_k} r_k(X,Y)^{\gamma_k} \tag{5}$$

As it can be seen from equation (5), mean, variance and cross-correlation are computed from the whole image at corresponding scale. α_k, $\{\beta_k\}_{k=1}^{K}$ and $\{\gamma_k\}_{k=1}^{K}$ are non-negative constants that

change based on the image scale. According to experiments of [13], $\alpha_k = 0.1333$, $\beta_1 = 0.0448$, $\beta_2 = 0.2856$, $\beta_3 = 0.3001$, $\beta_4 = 0.2363$, $\beta_5 = 0.1333$ and $\gamma_k = \beta_k$ for k=1,2, …, K are considered. Note that summation of β_k and γ_k must be one (i.e. $\sum_{k=1}^{K} \beta_k = 1$ and $\sum_{k=1}^{K} \gamma_k = 1$).

2.2.2 MS-SSIM values for our results

MS-SSIM is used to judge about structural similarity between two images (original and reconstructed images). Therefore, values closer to one show more similarity. Equation (5) was used along with the above mentioned values as equation's parameters to compute MS-SSIM.

3. Results and Discussion

The experimental video sequences in our task are *waterfall* and *pond*. Table 1 shows details of these videos. Note that for the first video we defined two masks in two different portions of the video based on the amount of variation in pixel information in that region.

Table 1. Specifications of the exprimental videos

Video	Number of Frames	Frame Rate	Number of Corrupted Frames
Waterfall 1	49	14.55	40
Waterfall 2	49	14.55	40
Pond	150	29.97	70

Fig. 4 depicts the inpainted frames corresponding to those shown in Fig. 2.

Table 2 shows overall MS-SSIM, min MS-SSIM and max MS-SSIM values for different reconstructed frames per video.

Table 2 ensures that the overall MS-SSIMs are 0.9854, 0.9943 and 0.9854, respectively that are very close to one. Therefore, the overall structure is preserved in this video.

Figs. 5 and 6 in section 3 give pictorial description about MS-SSIM values.

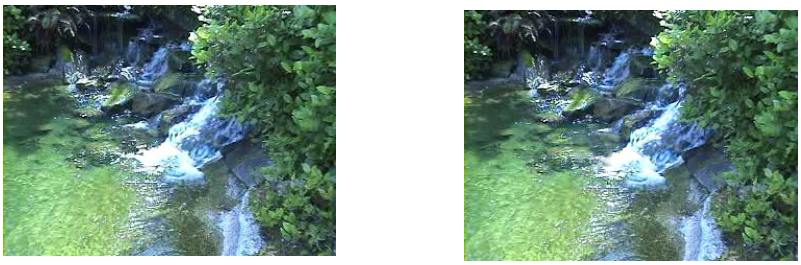
In Fig. 5 results of the proposed algorithm on *waterfall 1* for some random frames with corresponding MS-SSIM values are shown.

Fig. 6 shows the results of our method on *waterfall 2* for the same frames as in Fig. 5.

Fig. 7 compares the obtained results of our proposed method with the result of [9] on the *waterfall* video sequence. Comparison between columns of Fig. 7 shows that the results of the proposed algorithm are better than the results of [9]. Since we used a contour-based method for smoothing, there is no over-smoothing in our results. We only smooth the area near the boundary of reconstructed portion of the frame.

Fig. 8 shows the comparison of our approach with the one introduced in [9] on the pond video.

(a)



(b)

(c)

Fig. 4. Results of the video inpainting. Left column shows the original frames which their masks were shown in Fig. 2. Right column shows the corresponding inpainted frames.

Table 2. MS-SSIM for three reconstructed video sequences.

First Video	Second Video	Third Video	
0.9854	0.9943	0.9854	Overall MS-SSIM
0.9780	0.9868	0.9716	Min MS-SSIM
0.9942	0.9972	0.9926	Max MS-SSIM

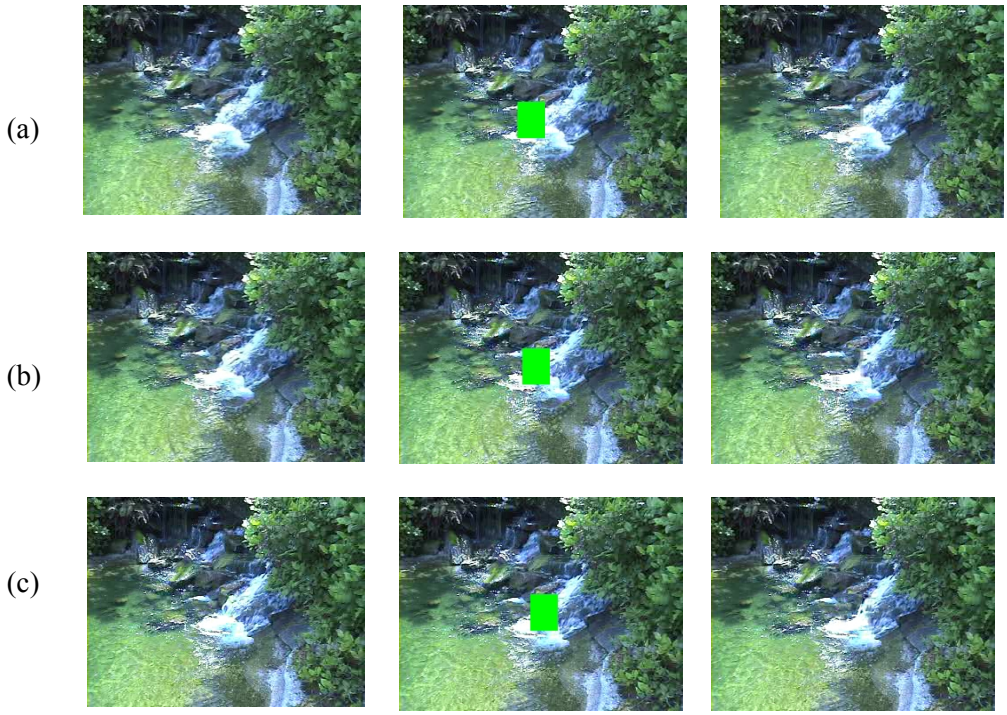


Fig 5. Results of the proposed video inpainting algorithm on video *waterfall 1*. (a) Frame 30 with MS-SSIM=0.9831, (b) frame 40 with MS-SSIM=0.9814, (c) frame 49 with MS-SSIM=0.9830. In this video the mask is moving slowly towards right of the scene.

In Fig. 5 the mask is moving slowly towards right of the scene. The region of the mask on (a) contains more calm water than the region of the mask on (b); as you can see it suddenly changes to clamorous water therefore, MS-SSIM decreases on (b) compared to (a). Yet, in later frames, there is no sudden change; changes are normal. As a result, the obtained model on (c) is more accurate compared to what was on (b) which ends to an increase in MS-SSIM.

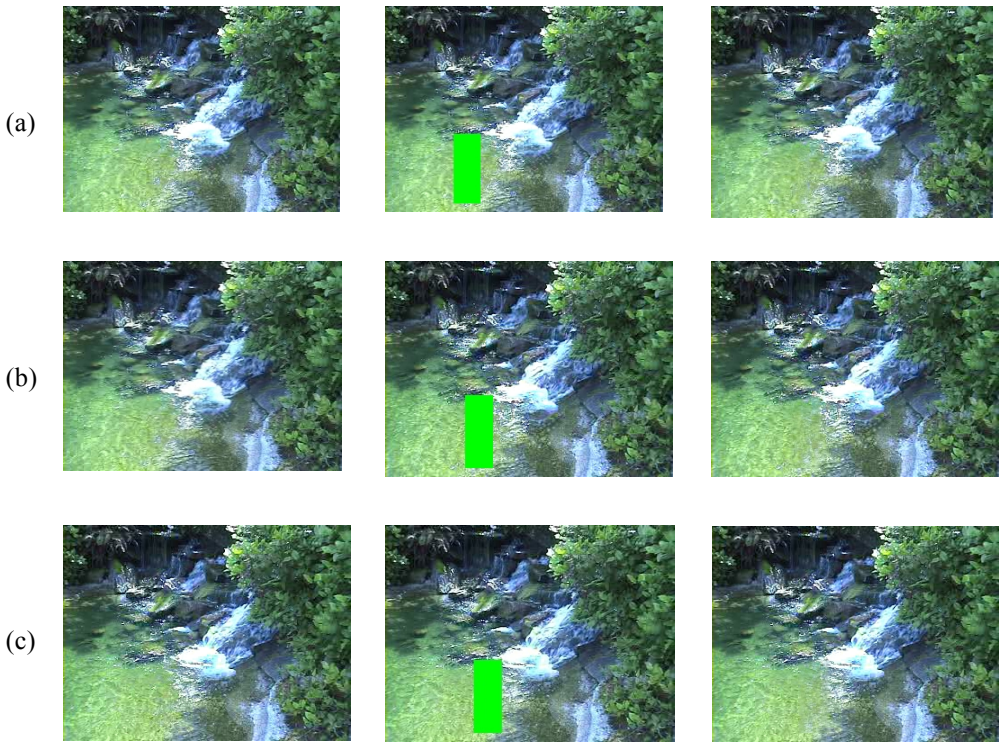


Fig 6. Results of the proposed algorithm on *waterfall 2*. (a) frame 30 with MS-SSIM=0.9954 (b) frame 40 with MS-SSIM=0.9929 and (c) frame 49 with MS-SSIM=0.9936. Here, in this video the mask is moving slowly towards right of the scene, too. Description about MS-SSIM values is similar to the one described for Fig. 5.

Fig. 7. Comparison of the results of [9] with the proposed algorithm (a) The proposed algorithm's result on "waterfall 1"; (b) The proposed algorithm's result on "waterfall 2"; (c) Results of the algorithm in [9]. No over-smoothing can be seen in our algorithm.

(a) (b) (c)

Fig. 8. Comparison of the results of [9] with those of the proposed algorithm (a) Original frame; (b) Results of the algorithm in [9]; (c) Results of the proposed algorithm. Structure is much more preserved in our method.

For comparison, the resulting videos can be viewed at:

- https://rapidshare.com/files/3596696068/waterfall_jumping_inpainted1.avi
- https://rapidshare.com/files/4008930424/waterfall_jumping_inpainted2.avi
- https://rapidshare.com/files/4274795887/pond_original_inpainted3.avi

4. Conclusion

In this paper, a video inpainting method was proposed to complete dynamic regions of corrupted video frames based on LDS. With respect to the damaged portion's coordinates, a dynamic texture model for that region is found. Then, using this model a new patch is generated corresponding to the damaged area. In order to decrease over-smoothing effect, a contour-based filtering method was applied to the boundary of the inpainted patch and its surrounding area in the original frame. In this study we have assumed there is enough information to build a model for texture (i.e. the target region never begins at early frames). Considering sever conditions such as moving camera introduces more challenging problems which we are working on.

Authors' contributions

All authors read and approved the final manuscript.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Bertalmio M, Bertozzi AL, Sapiro G, Navier-Stokes. Fluid Dynamics and Image and Video Inpainting. In Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition. 2001;1: 355-362.

[2] Cheung SS, Zhao J, Venkatesh M. Efficient Object-Based Video Inpainting. IEEE Int. Conf. Image Processing. 2006;705–708.

[3] Wexler Y, Shechtman E, Irani M. Space-Time Completion of Video. IEEE Trans. Pattern Analysis Machine Intelligence. 2007;29(3): 463-476.

[4] Venkatesh MV, Cheung SS, Zhao J. Efficient Object-Based Video Inpainting. Journal Of Pattern Recognition Letters. 2009;30(2): 168-179.

[5] Zhang Y, Xiao J, Shah M. Motion Layer Based Object Removal in Videos. In Proc. 7th IEEE Workshop On Applications Of Computer Vision. 2005;1: 516-521.

[6] Ding T, Sznaier M, Camps O I. A Rank Minimization Approach to Video Inpainting. in Proc. IEEE Int. Conf. Computer Vision. 2007;1-8.

[7] Soatto S, Doretto G, Wu Y N. Dynamic textures. in Proc. IEEE Int. Conf. Computer Vision. 2001;2: 439-446.

[8] Yuan L, Wen F, Liu C, Shum H Y. Synthesizing Dynamic Texture with closed-loop Linear Dynamic System. in Proc. European Conf. Computer Vision. 2004;2:603-616.

[9] Lin C W, Cheng N C. Video Background Inpainting Using Dynamic Texture Synthesis. In Proc. IEEE Int. Symposium on Circuitsand Systems. 2010;1559-1562.

[10] Ghanbari A, Soryani M. Contour-Based Video Inpainting. IEEE Iran. 7th Machine Vision and Image Processing. 2011; 1-5.

[11] Wang Z, Bovik A C, Sheikh H R, Simoncelli E P. Image Quality Assessment : From Error Visibility to Structural Similarity. IEEE Trans. Image Processing. 2004;13(4): 600-612.

[12] Rouse D M, Hemami S S. Understanding and Simplifying the Structural Similarity Metric. 15th IEEE Int. Image Processing. 2008; 1188-1191.

[13] Wang Z, Simoncelli E P, Bovik A C. Multi-Scale Structural Similarity for Image Quality Assessment. in Proc. 37th IEEE Asilomar Conf. on Signals, Systems and Computers. 2003;2: 1398-1402.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
www.sciencedomain.org/review-history.php?iid=615&id=6&aid=5512